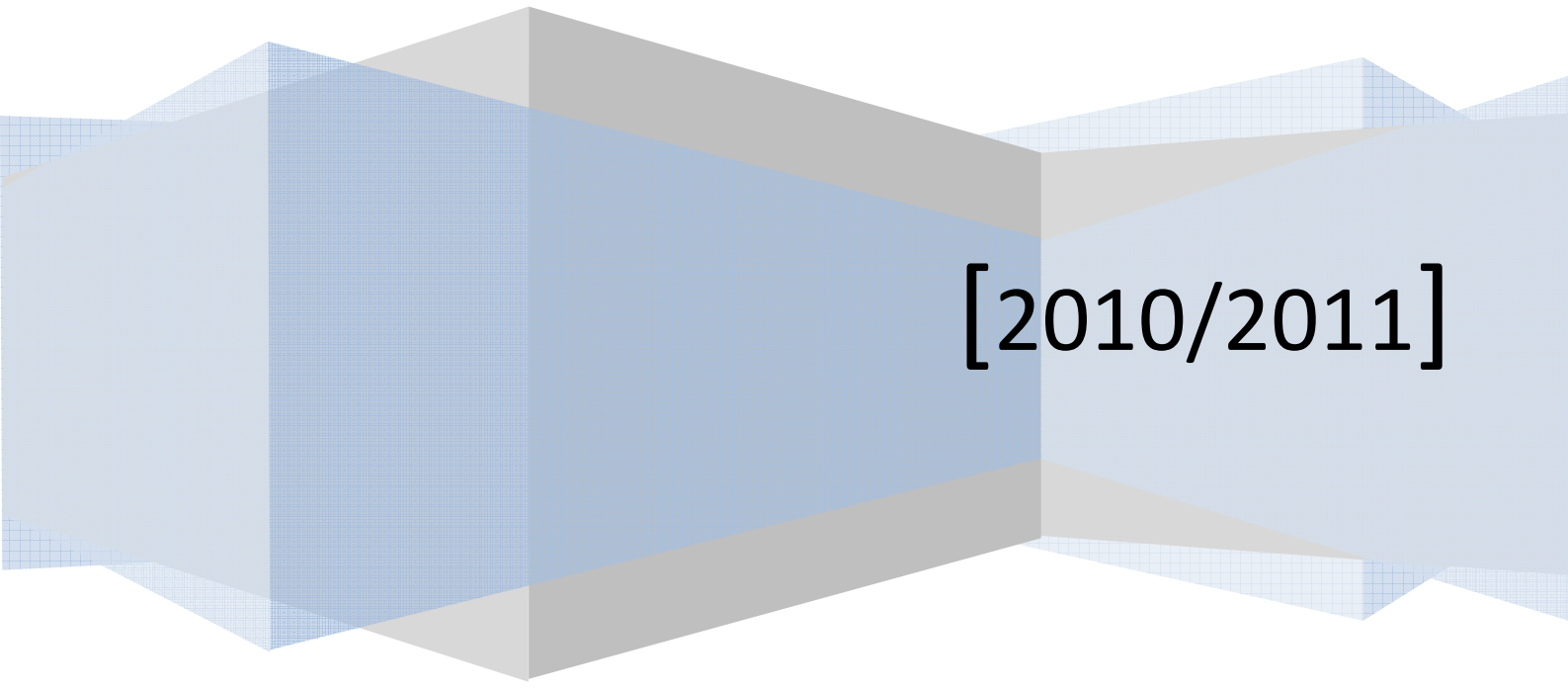
Proyecto de Fin de Máster

Aceleración del algoritmo Smith-Waterman mediante dispositivos reconfigurables.

Autor: Roberto Salvador Gradaille
[rsalvadorg@gmail.com]

Tutor: Gustavo Sutter
[gustavo.sutter@uam.es]



[2010/2011]

RESUMEN

El algoritmo Smith-Waterman [1] [2] es uno de los algoritmos más importantes de alineación local de secuencias biológicas, como ADN (Acido desoxiribonucleico) o ARN (Acido ribonucleico), es decir determina si dos secuencias biológicas tienen algún fragmento en común, o son de una gran similitud biológica teniendo en cuenta las posibles mutaciones, inserciones o eliminaciones de elementos dentro de una cadena. Por este motivo este algoritmo es usado en diferentes ámbitos bioinformáticos con el fin de localizar alineamientos de carácter local, siendo el caso de uso más importante su implicación dentro del programa BLAST (Basic Local Alignment Search Tool) [3].

BLAST es un programa informático utilizado para realizar alineamientos de carácter local entre secuencias biológicas, siendo capaz de encontrar en una base de datos de secuencias el conjunto de secuencias similares a una secuencia objeto mediante técnicas heurísticas, determinando el grado de similitud entre cada secuencia obtenida y la secuencia objeto. Por este motivo es el principal programa de alineación de secuencias biológicas utilizado por la comunidad biológica internacional.

Tanto BLAST como la mayoría de las variaciones del mismo, utilizan el algoritmo Smith-Waterman para determinar la zona donde se produce una mayor similitud entre dos cadenas y a partir de esta información realizar la fase de extensión del BLAST con el fin de determinar la similitud de ambas cadenas a nivel global.

Debido al gran uso que están teniendo en la actualidad los sistemas basados en BLAST, es de vital importancia obtener un rendimiento óptimo de los mismos, pues cada vez es mayor el número de secuencias biológicas a analizar así como el número de secuencias almacenadas en las bases de datos, lo que hace que el tiempo de ejecución de un alineamiento de secuencias biológicas se esté viendo incrementado constantemente. En este trabajo se trata de analizar las posibles mejoras que presenta el algoritmo Smith-Waterman al estar implementado en un hardware específico, con el fin de poder obtener un sistema BLAST cuya velocidad de ejecución sea considerablemente superior a la que obtienen los sistemas BLAST existentes actualmente y poder de este modo ofrecer una solución real a la comunidad científica que día a día tiene que sufrir los actuales tiempos de ejecución de los sistemas BLAST para obtener la información necesaria para su trabajo diario.

La aproximación de este trabajo es la utilización de aceleradores hardware basados en FPGAs (Altera Stratix 3) conectados al sistema de computo a través del bus del sistema, en este caso *Front-side bus* (FSB) de Intel, usando módulos “in-socket” y programados desde lenguajes de alto nivel (Impulse-C).

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi tutor Gustavo Sutter su apoyo constante durante estos dos años de Máster. Gracias a él me he introducido en un mundo completamente desconocido para mí y de gran interés tecnológico como es el High Performance Reconfigurable Computing.

Agradecer del mismo modo el apoyo recibido por todos los miembros del grupo de redes y computación de alto rendimiento (HPCN) de la Universidad Autónoma de Madrid, los cuales me han facilitado toda la ayuda posible así como la posibilidad de asistir a diferentes reuniones relacionadas con los temas tratados en el presente trabajo. Así mismo me gustaría agradecer a Pixelware S.A. todas las facilidades recibidas durante estos dos años, sin las cuales habría sido imposible compaginar el máster con la vida laboral.

Finalmente, quiero hacer una especial mención a mi familia, a mi madre Lola, mi padre Narciso, mi hermano Fernando, y especialmente a Isa. Gracias por ayudarme a seguir adelante durante todo este tiempo.

Muchas gracias a todos por vuestro apoyo.

Roberto Salvador Gradaille.

INDICE DE CONTENIDOS

1. INTRODUCCIÓN AL PROYECTO

1.1. Motivación

Actualmente en el mundo de la biología uno de los mayores problemas existentes es la gran cantidad de información disponible y por tanto la dificultad intrínseca de manejar dicha cantidad de datos con el fin de poder analizar dichos datos de manera global y establecer relaciones entre ellos. Por este motivo la bioinformática, aplicación de las tecnologías de la información a la gestión y análisis de información biológica, ha sufrido un gran crecimiento durante los últimos años desarrollando soluciones informáticas a problemas presentes en la actividad diaria de la comunidad científica.

Un claro ejemplo de este problema es la alineación de secuencias de carácter biológico como pueden ser cadenas de proteínas, cadenas de ADN, cadenas de ARN, aminoácidos, etc. Todas estas cadenas llenas de información han de ser procesadas, entre otras formas, mediante comparaciones con cadenas similares ya reconocidas anteriormente, este proceso es conocido como alineamiento de secuencias biológicas. Debido a la gran cantidad de secuencias reconocidas y a la longitud de las mismas, esta tarea ha requerido históricamente de un alto trabajo computacional. Este problema se ha visto continuamente agravado por el crecimiento constante de las bases de datos de secuencias reconocidas, tal y como puede verse en el ejemplo del GenBank del NCBI representado en Fig1, presentándose como única solución la mejora de los sistemas encargados de dicha carga computacional. Esta solución se ha visto sobrepasada por el ritmo de crecimiento de las bases de datos siendo necesario un análisis más profundo del problema con el objetivo de encontrar nuevas soluciones que reduzcan los tiempos de computación de estos sistemas.

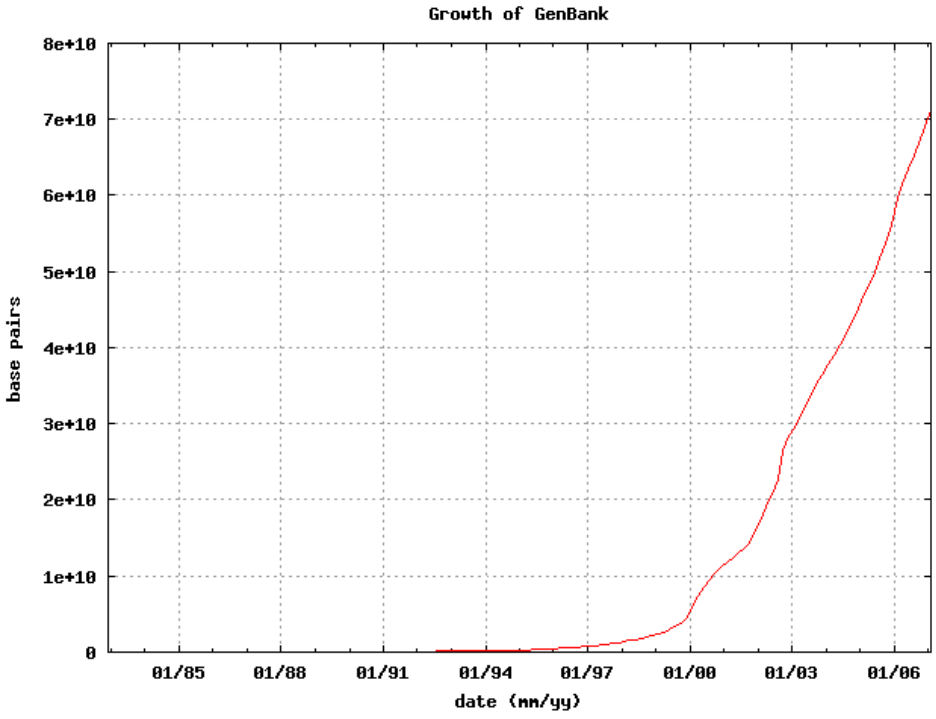


Fig. 1 Evolución del tamaño del GenBank

1.2.Objetivo

Este trabajo analiza, estudia y trata de resolver el problema existente actualmente en el mundo de la bioinformática con el tiempo de computación necesario para obtener los alineamientos locales entre un conjunto de secuencias a analizar y una base de datos de secuencias biológicas reconocidas.

Una de las mayores complicaciones existentes a la hora de afrontar este problema es que a diferencia de otros sistemas en este caso no existe una solución determinista, si no que el software empleado actualmente es el patrón de ejecución válido. Esto se debe a que la comunidad biológica considera estos resultados heurísticos como los resultados óptimos, por lo que cualquier nuevo algoritmo de alineamiento, así como las modificaciones realizadas sobre el software existente, tienen como objetivo obtener los mismos resultados que se obtienen actualmente, o unos resultados con un gran grado de similitud, en un tiempo de ejecución inferior al actual.

El sistema de referencia actual es el BLAST del NCBI, así como todas las modificaciones realizadas sobre el mismo con el fin de particularizar el programa para un determinado conjunto de secuencias biológicas tales como proteínas, aminoácidos, etc.

Tal y como se analizará más adelante, una de las partes más costosas a nivel computacional es la ejecución del algoritmo Smith-Waterman, debido principalmente a la gran cantidad de veces que es ejecutado en el programa. Siguiendo la línea marcada por trabajos anteriores como son [4] y [5], en este trabajo se van a estudiar las alternativas existentes en hardware reconfigurable a la ejecución software del algoritmo Smith-Waterman, para posteriormente presentar e implementar una nueva alternativa de aceleración mediante hardware reconfigurable del algoritmo Smith-Waterman con el fin de que este pueda ser integrado en un sistema BLAST completo.

1.3.Descripción del trabajo

El presente trabajo de fin de máster, ha quedado dividido en tres actividades claramente diferenciadas, a través de las cuales se pretende obtener por un lado un conocimiento pormenorizado del algoritmo Smith-Waterman, de su integración con BLAST y de las alternativas hardware existentes actualmente, con el fin de poder realizar una análisis de requisitos, un diseño basado en dicho análisis y finalmente una implementación del algoritmo Smith-Waterman en codiseño hardware/software que obtenga un mayor rendimiento que la versión tradicional software.

Actividad 1. Algoritmo Smith-Waterman, su implicación en BLAST y uso del mismo en la actualidad. Durante esta etapa del proyecto se ha realizado un proceso de investigación a nivel conceptual del programa BLAST y del algoritmo Smith-Waterman como parte de este.

Actividad 2. Alternativas existentes al algoritmo Smith-Waterman mediante dispositivos reconfigurables. Estado del arte del algoritmo Smith-Waterman mediante arquitecturas basadas en hardware reconfigurable.

Actividad 3. Smith-Waterman en dispositivos reconfigurables mediante registros de rotación.

- Análisis general del sistema.
- Implementación del sistema sobre un dispositivo concreto.
- Análisis de rendimiento.

1.4.Estructura de la memoria

El presente documento queda estructurado en seis capítulos, de los cuales el primero de ellos contiene la introducción al proyecto, seguido de tres capítulos uno por cada actividad detallada anteriormente. En el quinto capítulo se muestran las conclusiones obtenidas a la finalización del proyecto realizado y finalmente en el sexto y último capítulo se presentan las referencias bibliográficas utilizadas.

2. ALGORITMO SMITH-WATERMAN, SU IMPLICACIÓN EN BLAST Y USO DEL MISMO EN LA ACTUALIDAD

2.1. Alineamiento de secuencias

Tal y como se indica en [6] el alineamiento de secuencias biológicas es la comparación de dos secuencias o cadenas de ADN, ARN, proteínas, aminoácidos u otras estructuras primarias, con el fin de encontrar sus zonas de similitud que podrían indicar relaciones funcionales, estructurales o evolutivas entre los genes o proteínas analizados. En cadenas con antecesores comunes los elementos diferentes en el análisis se consideran mutaciones puntuales y los huecos son considerados como eliminaciones o inserciones de elementos en una u otra cadena.

Las técnicas de alineamiento de secuencias biológicas se utilizan principalmente en el campo de la ingeniería genética a la hora de estudiar evoluciones de los organismos, pero adicionalmente son utilizados en otros campos de investigación como por ejemplo el procesamiento de lenguaje natural.

Dentro de los alineamientos de secuencias se pueden diferenciar dos tipos claramente diferenciados, alineamientos globales y alineamientos locales.

- Los alineamientos globales son aquellos que tratan de alinear dos secuencias de manera global, esta técnica ofrece unos resultados muy interesantes cuando las cadenas que se analizan presentan un alto grado de similitud, tanto en la longitud de las mismas como en el contenido de estas. El algoritmo más común utilizado para realizar alineamientos globales el algoritmo de Needleman-Wunsch.

Ejemplo de alineaminto global.

```
aaagcggaagtcacag
||.||.|||| |.||
aaggctgaagt-atag
```

- Los alineamientos locales tratan de obtener regiones comunes entre ambas secuencias independientemente de la similitud encontrada en los elementos adyacentes a dichas regiones. Este tipo de alineamiento es utilizado en cadenas con baja similitud pero que pueden contener regiones similares. El algoritmo más común para realizar alineamientos globales el algoritmo Smith-Waterman.

Ejemplo de alineamiento local.

```
aaagcggaagtcacag
......|||| ....
aaggctgaagt-atag
```

En función del objetivo deseado y de las cadenas a analizar resulta de mayor interés un tipo u otro de alineamiento, incluso en cadenas de muy alta similitud no existe diferencia entre un tipo de alineamiento y otro. En la actualidad, de cara a la comunidad bioinformática, resulta de mayor interés la búsqueda de alineamientos de secuencias de manera local.

2.2.BLAST, el programa de alineamiento de secuencias

BLAST (Basic Local Alignment Search Tool) es el programa de alineamiento local de secuencias biológicas por excelencia y puede ser considerado como una evolución del algoritmo Smith-Waterman o más bien capa encima del algoritmo Smith-Waterman, la primera versión de BLAST desarrollada en 1990 por un equipo de científicos estadounidenses [3] pertenecientes en su mayoría al NCBI (National Center for Biotechnology Information). Esta primera versión del programa simplemente realizaba alineamientos de secuencias biológicas sin huecos limitando la búsqueda de secuencias sustancialmente. Posteriormente se han ido desarrollando nuevas versiones del programa que resuelven problemas concretos como son BLASTp, BLASTn, TBLASTn, TBLASTx, o BLAST 2.0 (a partir de ahora BLAST) [7]. Esta última modificación del programa es la que se utiliza actualmente y se conoce comúnmente como BLAST, en ella se permite realizar las mismas búsquedas que el la primera versión del programa con la diferencia que se permiten inserciones y eliminaciones en las secuencias analizadas, ofreciendo una amplia variedad de modificaciones no disponibles anteriormente.

La mayoría de las herramientas vistas anteriormente se pueden encontrar a disposición del usuario, tanto de manera on-line como descargable, en la página web del NCBI [8]. La versión on-line tiene la gran ventaja de utilizar las bases de datos mantenidas por el NCBI, pero por el contrario no permite trabajos con carga masiva al ser un recurso compartido por toda la comunidad científica. Todo el código se encuentra disponible en la web del NCBI, pudiendo realizar sobre aquellas modificaciones que cada investigador considere oportunas, dando lugar a nuevas versiones del programa.

De cara a realizar una mejora de rendimiento en el programa BLAST es importante conocer detalladamente el funcionamiento del mismo, por este motivo se presentan a continuación las tres etapas que componen la ejecución del programa, la primera etapa denominada etapa de ensemillado, la segunda etapa de extensión y la tercera etapa de evaluación.

- Ensemillado. El objetivo de esta primera etapa del algoritmo es la de realizar una búsqueda en la base de datos de todas aquellas secuencias que puedan tener alguna similitud con la secuencia objeto, para lo cual se realizan las siguientes fases.

a) La cadena a analizar se divide en todas las sub cadenas existentes de tamaño W tal y como se muestra en Fig2, obtenida de [9].

b) Con cada cadena de la fase a se generan todos los posibles vecinos de la misma. Para ello se obtienen todas las posibles combinaciones de W letras contenidas en el alfabeto utilizado, y de estas se seleccionan aquellas cuya distancia sea menor que A frente a la subcadena en cuestión, finalmente se puntúan estas cadenas mediante una matriz de sustitución y se seleccionan aquellas que superen un valor determinado T.

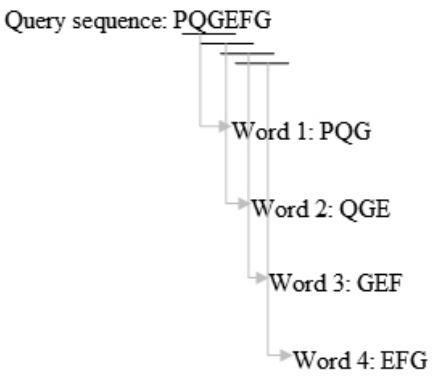


Fig. 2 División en subcadenas

c) Para finalizar la etapa de ensemillado todas las cadenas de tamaño W obtenidas en la fase b) del ensemillado son buscadas literalmente en la base de datos, obteniendo todas aquellas secuencias que contengan alguna de estas cadenas de tamaño W.

Los valores W, A y T utilizados en la fase de ensemillado son parámetros de entrada del programa y en función de ellos se determinada tanto el grado de fiabilidad de los resultados como el tiempo de ejecución necesario para la obtención de los mismos.

- Extensión. Una vez seleccionado el conjunto de cadenas de la base de datos que van a ser analizadas, se ejecuta el algoritmo Smith-Waterman comparando la secuencia objeto con todas ellas, el algoritmo Smith-Waterman se detallará en el siguiente punto del documento. La ejecución del algoritmo genera una matriz por cada par de cadenas, el valor máximo de dicha matriz determinará si el alineamiento en cuestión supera un valor umbral determinado, en caso negativo el alineamiento queda descartado y en caso contrario se comienza un proceso denominado Backtrace, del alto coste computacional, mediante el cual se obtiene el alineamiento local de ambas cadenas con huecos. Finalmente con este alineamiento local obtenido se realiza una extensión en ambas direcciones del mismo hasta que valor del alineamiento desciende un valor determinado respecto al valor máximo obtenido previamente en el algoritmo Smith-Waterman.

- Evaluación. Finalmente se evalúan todos los alineamientos obtenidos descartando aquellos en los que una misma parte de la secuencia objeto se ha alineado con diferentes segmentos de una misma secuencia de la base de datos. Todos aquellos alineamientos obtenidos son denominados High Score Pairs o HSPs.

El éxito de BLAST se basa principalmente en su tiempo de ejecución más que en la exactitud de sus resultados ya que este basa los alineamientos obtenidos en una primera búsqueda heurística, por contra el algoritmo Smith-Waterman es el algoritmo óptimo de alineación de secuencias biológicas pero el coste computacional del mismo hace imposible la ejecución de este frente a una base de secuencias completa. Por este motivo el programa BLAST trata de hacer una preselección de cadenas objetivo para reducir el número de cadenas a alinear mediante Smith-Waterman y por tanto el tiempo de ejecución del programa, este proceso se realiza durante la fase de ensemillado donde se obtiene un subconjunto de cadenas de la base de datos, a priori con mayor probabilidad de obtener una alta puntuación en el alineamiento.

Pese a las mejoras de rendimiento obtenidas por BLAST frente al método clásico del algoritmo Smith-Waterman simple, el tamaño de las bases de datos de secuencias hace que los resultados obtenidos en la fase de ensemillado sean demasiado grandes y por tanto la ejecución de todas estas cadenas en el algoritmo Smith-Waterman produzca un gran tiempo de computación.

2.3. Smith-Waterman

El algoritmo Smith-Waterman data del año 1981 y fue desarrollado por los investigadores Temple F. Smith y Michael S. Waterman, este algoritmo publicado en [2] y [1] está basado en el algoritmo de Needleman-Wunsch [10] de alineamiento de secuencias globales mediante una matriz calculada de manera iterativa, de este modo ambos algoritmos están considerados como algoritmos de programación dinámica.

El algoritmo Smith-Waterman es considerado actualmente como el algoritmo óptimo de alineamiento de secuencias biológicas de manera local en función del sistema de puntuación utilizado, por el contrario presenta un gran coste computacional y un gran consumo de recursos del sistema, ambos factores directamente proporcionales a los tamaños de las cadenas a analizar siendo el coste computacional del mismo **O(nm)** donde *n* y *m* son los tamaños de las cadenas a analizar. Por este motivo el uso de este algoritmo como sistema de búsqueda de alineamientos locales deja de ser viable cuando los tamaños de las cadenas son valores grandes.

El proceso del algoritmo se puede separar en dos fases claramente diferenciadas, la primera de ellas es la generación de la matriz de resultados y la segunda es el denominado proceso de BackTrace desde el punto donde se encuentre el valor máximo de la matriz.

La matriz obtenida en la primera fase del algoritmo y en la cual se basa el proceso de BackTrace del algoritmo viene definida en [1] cómo:

$$H_{ij} = max\left\{H_{i-1,j-1} + s(a_i, b_j), \max_{k\geq 1}\{H_{i-k,j} - W_k\}, \max_{l\geq 1}\{H_{i,j-l} - W_l\}, 0\right\}$$

$$1 \leq i \leq n \; y \; 1 \leq j \leq m$$

$$H_{k0} = H_{0l} = 0 \; para \; 0 \leq k \leq n \; y \; 0 \leq l \leq m$$

Siendo n y m las longitudes de las cadenas a analizar.

Analizando detalladamente la formula se tiene que $H_{i-1,j-1} + s(a_i, b_j)$ representa una modificación en la cadena a en la posición i al elemento en la posición j de la cadena b, $\max_{k\geq 1}\{H_{i-k,j} - W_k\}$ representa que el elemento a_i es parte de una inserción o eliminación de longitud k, del mismo modo que $\max_{l\geq 1}\{H_{i,j-l} - W_l\}$ representa una inserción o eliminación de longitud l en b, finalmente se añade el posible valor 0 para evitar posibles valores negativos en la matriz.

La gran ventaja que presenta este algoritmo es la posibilidad de definir un sistema de puntuación que determine los valores de las variables W_k y $s(a_i, b_j)$ los cuales determinarán la precisión en la búsqueda, inicialmente en [1] estos parámetros se definieron como $W_k = 1 + k / 3$ y $s(a_i, b_j) = 1$ si $a_i = b_j$ y $s(a_i, b_j) = -1 / 3$ en caso contario. En versiones posteriores el valor de $s(a_i, b_j)$ se obtiene de una matriz de sustitución especifica, existiendo diferentes posibles matrices [11] de sustitución siendo las más comunes las matrices BLOSUM y la matriz PAM. Todas estas matrices especifican el valor que se obtiene tanto en la sustitución de un elemento por otro de la matriz en la cadena, así como el valor que se obtiene al mantenerse constante el elemento de la cadena. Otra modificación existente en versiones posteriores del algoritmo Smith-Waterman es la posibilidad de asignar una penalización diferente a la apertura de un nuevo hueco que a un elemento que amplía el tamaño de un hueco ya existente.

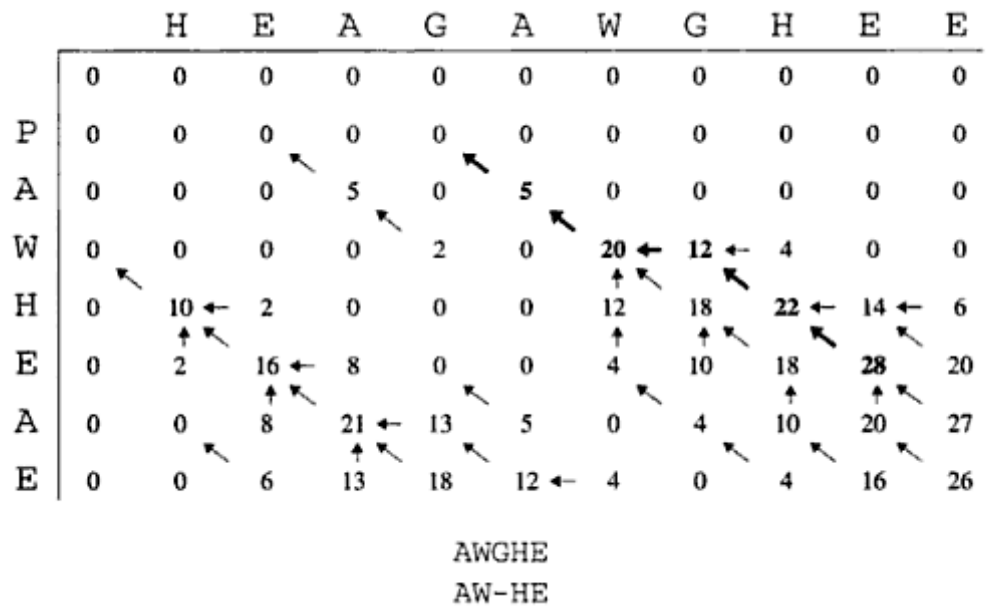


Fig. 3 Matriz Smith-Waterman

Una vez que se ha obtenido la matriz de resultados del algoritmo se localiza el valor máximo existente en la misma y desde este punto se realiza el proceso de BackTrace mediante el cual se obtendrá en alineamiento final. En el proceso de BackTrace se realiza un recorrido desde el valor máximo de la matriz hasta que se llegue a un valor igual a 0, en cada punto de dicho recorrido se realizará un paso hacia el valor previo que ha propiciado la obtención del valor actual. El proceso de BackTrace puede verse en la fig. 3, obtenida de [12], la cual se obtiene de alinear las secuencias HEAGAWGHEE Y PAWHEAE, mediante el algoritmo Smith-Waterman con una penalización por hueco de 8 y utilizando como matriz de pesos la matriz BLOSUM45. En esta matriz se puede observar como el camino de BackTrace comienza en el valor máximo, en este caso 28, y mantiene almacenados todos los posibles caminos que se han generado para de este modo poder recorrer el camino desde el valor máximo, el cual queda marcado en negrita en la fig. 3.

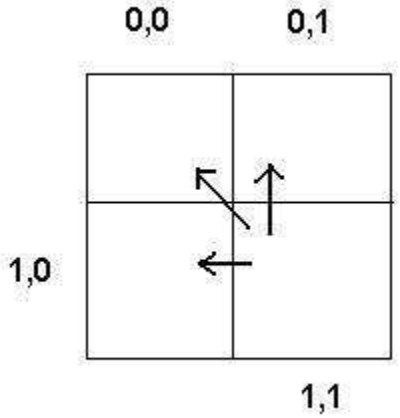


Fig. 4 Dependencia de datos

Tal y como se puede observar en el ejemplo la inserción de un hueco en la cadena inferior equivale con el desplazamiento lateral, el desplazamiento vertical del recorrido equivale a un hueco en la cadena superior y la diagonal equivale a alinear los valores de dichas posiciones sean o no el mismo valor, pese a que este último caso no se da en el ejemplo.

La dependencia de datos presentada en el algoritmo Smith-Waterman tal y como se puede ver en la fórmula de cálculo de elementos de la matriz H, así como en el ejemplo planteado anteriormente, indica que un determinado valor de la matriz depende única y exclusivamente del elemento a la izquierda del mismo, del elemento superior al mismo y del elemento situado en la diagonal superior izquierda de este, esta dependencia puede verse representada en la Fig. 4. Este detalle es uno de los fundamentos de los sistemas analizados en los puntos 3 y 5 del presente documento.

Debido a la estructura del algoritmo, para obtener el alineamiento local de dos secuencias de tamaños *n* y *m* respectivamente se requiere mantener almacenado en memoria un estructura de *n* x *m* elementos, además se requiere realizar *n* x *m* cálculos donde todos y cada uno de ellos necesitan del acceso a otra estructura de memoria, que será la matriz de pesos en cuestión. Todo este conjunto de datos en memoria y capacidad de proceso es sin tener en cuenta el proceso de BackTrace, el cual precisa de un gran coste computacional ya que debe de analizar por cada elemento del camino las tres posibles opciones existentes y decidir de cuál de las tres opciones se ha obtenido su valor para poder continuar recursivamente con el algoritmo.

3. ALTERNATIVAS EXISTENTES AL ALGORITMO SMITH-WATERMAN MEDIANTE DISPOSITIVOS RECONFIGURABLES

3.1.Introducción

El algoritmo Smith-Waterman [2] está basado en programación dinámica y obtiene el alineamiento local óptimo entre dos secuencias de carácter biológico, por el contrario este algoritmo tiene un gran coste computacional y de memoria como ya se ha podido intuir anteriormente gracias a la estructura del mismo. Por este motivo este algoritmo se ha tratado de acelerar por diferentes métodos a lo largo de la historia con el fin de poder obtener alineamientos óptimos en un tiempo de ejecución pequeño.

Se han utilizado métodos heurísticos con el fin de reducir el número de secuencias a analizar por el algoritmo, en este caso en lugar de reducir el tiempo de computación del algoritmo se reduce el número de veces que ha de ser ejecutado para encontrar una solución frente a un amplio conjunto de secuencias. El ejemplo más claro de este tipo de alternativa es el BLAST [3].

Otra metodología utilizada es la aceleración del algoritmo mediante métodos conservativos, los cuales no interfieren en la ejecución del mismo. En este conjunto se pueden englobar soluciones que ha día de hoy se encuentran accesibles fácilmente como pueden ser los sistemas con varios núcleos de procesamiento.

Otra alternativa comúnmente utilizada para la aceleración de procesos, y que también ha sido utilizada para la aceleración del algoritmo Smith-Waterman [2] es la computación distribuida, así como la computación distribuida mediante GPUs [13]. En este conjunto se encuentra por ejemplo el proyecto Smith-Waterman CUDA [14] que ejecuta el algoritmo obteniendo un factor de beneficio de las tarjetas NVIDIA asociadas al programa CUDA.

Finalmente existe otra alternativa que es la aceleración de algoritmos mediante hardware reconfigurable, esta alternativa denominada HPRC (High Performance Reconfigurable Computing) también ha sido empleada como alternativa al problema de rendimiento existente en el algoritmo Smith-Waterman. Los sistemas HPRC se basan en dispositivos reconfigurables denominados FPGAs los cuales pueden ser configurados mediante software para que realicen una determinada labor a nivel hardware, con las mejoras de rendimiento asociadas a la ejecución hardware. En el presente apartado se realiza un estado del arte de las diferentes alternativas existentes actualmente al algoritmo Smith-Waterman mediante dispositivos reconfigurables, realizando un análisis y exposición de las mismas con el fin de conocer los avances existentes actualmente en este campo de la supercomputación reconfigurable y poder utilizarlos en la realización del presente proyecto.

A continuación se presentan aquellas modificaciones al algoritmo Smith-Waterman haciendo usos de sistemas basados en FPGAs tanto a nivel comercial como a nivel investigación.

3.2.Smith-Waterman mediante array sistólico

Debido a la dependencia de datos existente en el algoritmo Smith-Waterman el uso de arrays sistólicos es una de las alternativas más utilizadas. Para comprender estos sistemas es necesario entender en primer lugar el concepto de array sistólico.

- Array Sistólico: Una red sistólica es una red formada por unidades de procesamiento independientes en la que dichas unidades realizan su función y pasan los datos a los siguientes elementos de la red produciendo la reacción de estos ante la llegada de nueva información a procesar [15]. Se conoce como array sistólico a aquella red sistólica con forma matricial o de array. El uso de estas estructuras es comúnmente utilizado a la hora de realizar pipelines en computación, debido al paralelismo intrinseco de las mismas tal y como se puede observar en fig 4, $PE_{1,1}$ es el primer elemento que genera un resultado, en la siguiente etapa los elementos $PE_{1,2}$ y $PE_{2,1}$ generaran un resultado dependiente de los datos ofrecidos anteriormente por $PE_{1,1}$, este proceso se repite de manera continuada hasta que todos los elementos de la matriz han producido un resultado.

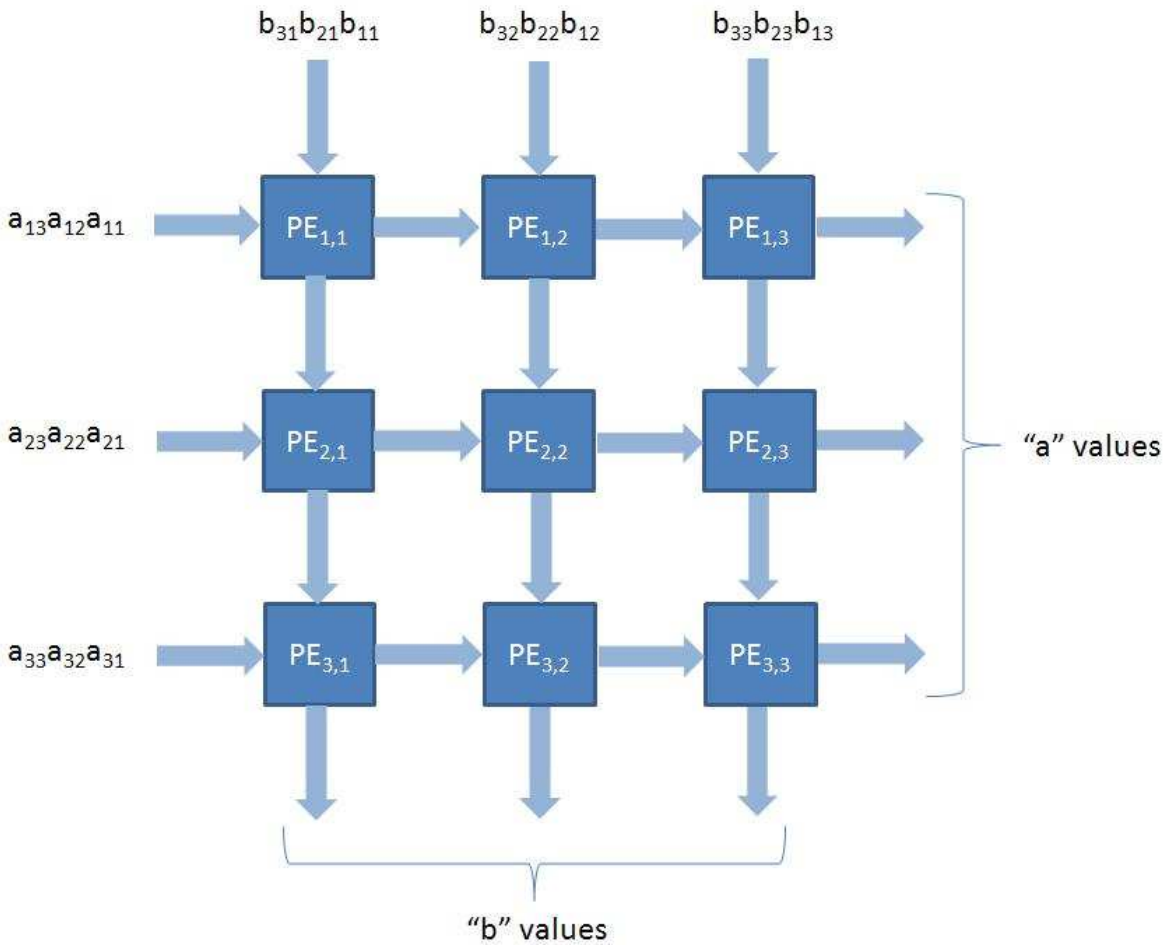


Fig. 5 Array Sistólico

En la Fig. 5 se muestra un ejemplo de array sistólico convencional, para la implementación del algoritmo Smith-Waterman mediante un array sistólico es necesario añadir conexiones diagonales entre las unidades de procesamiento básicas debido a la dependencia de los datos existente en el algoritmo.

Las diferencias entre los diferentes sistemas existentes basados en arrays sistólicos como son [4], [16], [17] y [18], se centran principalmente en las diferentes implementaciones de las unidades básicas de procesamiento, la inclusión del paralelismo y el procesamiento de los datos.

La mayoría de estas implementaciones presentan unas grandes limitaciones en lo que se refiere al tamaño de la cadena a analizar y principalmente en los tipos de componentes analizables, tan solo [18] permite el análisis de proteínas, realizando en resto un análisis de ADN lo cual permite simplifica en gran medida el diseño y por tanto el área utilizada.

3.3.Copacobana 5000 Smith-Waterman

Esta implementación del algoritmo Smith-Waterman utiliza el sistema reconfigurable masivo Copacobana 5000 [19], el cual dispone de un total de 18 slots con 16 FPGAs en cada uno de ellos. Esta implementación trata de descartar aquellos alineamientos que no superen un valor umbral dado, para ello utiliza todos los dispositivos disponibles en el sistema y en cada uno de ellos utiliza un array sistólico para realizar de forma paralela el cálculo de diagonales estudiado en el apartado 3.2 del presente documento.

Los resultados presentados en [19] muestran una aceleración de 750x, aunque esta aceleración se da en unos términos relativos que a continuación se detallan:

- El Smith-Waterman basado en Copacabana 5000 solamente es capaz de analizar alineamientos de ADN, lo que delimita los posibles valores a los cuatro componentes del ADN. Reduciendo los requisitos de área en gran medida.
- La aceleración obtenida es haciendo uso de 128 FPGAs en paralelo por lo que la aceleración será mucho mayor que en sistemas con uno solo dispositivo reconfigurable.
- Al ser un sistema de corte por valor umbral, la lógica indica que el sistema dejará de procesar al llegar al valor umbral dado y dicho valor umbral no está siendo indicado en [19]. Esto hace pensar que el valor umbral utilizado está siendo un valor tal que ninguna secuencia será procesada en software.

Como aspecto claramente positivo del sistema indicar la no limitación de tamaño de las secuencias ya que todas las FGPAs disponibles están conectadas entre sí formando un gran array sistólico y por tanto permitiendo el análisis de múltiples alineamientos de tamaño reducido o bien el análisis de un gran alineamiento el cual no estaría disponible en otros sistemas similares.

3.4.Banded Smith-Waterman para Mercury BLASTP

A pesar de que esta implementación no es una implementación del algoritmo Smith-Waterman tradicional, si no que es una modificación del mismo de cara a ser utilizado como parte del Mercury BLASTP [5]. Mercury BLASTP es una versión de BLASTP para dispositivos reconfigurables la cual obtiene unos resultados muy similares a la versión software de BLASTP.

La técnica utilizada es muy próxima al array sistólico de forma teórica, por el contrario utiliza una banda de cálculo de celdas capaz de almacenar las dos últimas iteraciones calculadas, de este modo se pasan los datos por estas celdas de tal forma que en cada iteración se calcule una nueva diagonal.

Una de las principales características del sistema es el agrupamiento de secuencias de pequeño tamaño, reduciendo de este modo el número de pasadas sobre la base de datos a consultar.

4. APROXIMACIÓN DE SMITH-WATERMAN PARA BLAST EN DISPOSITIVOS RECONFIGURABLES MEDIANTE ROTACIÓN

4.1.Introducción

Tal y como se ha visto a lo largo del presente documento el principal problema del algoritmo Smith-Waterman [2], [1] es el tiempo de ejecución del mismo debido a las repetidas veces que ha de ejecutarse la operación de cálculo de una celda de la matriz, motivo por el cual sistemas como BLAST [3] son más utilizados en la actualidad. Aun así el programa BLAST [3] realiza una modificación del algoritmo Smith-Waterman [2], [1] en su fase final por lo que la capacidad de BLAST [3]queda limitada en gran medida al rendimiento ofrecido por el algoritmo Smith-Waterman.

Debido a la importancia que supone el consumo de área en sistemas basados en FPGA es complicado almacenar toda la matriz necesaria para el cálculo del algoritmo Smith-Waterman [2], [1], por este motivo se propone una posible alternativa al algoritmo Smith-Waterman [2], [1] orientada a sistemas de tipo BLAST [3] con el fin de limitar el número de veces que ha de ejecutarse el algoritmo Smith-Waterman [2], [1] completo. En esta alternativa tan solo se almacena una pequeña cantidad de información básica para determinar si un alineamiento supera un determinado valor umbral.

4.2.Arquitectura del sistema BLAST

En sistemas BLAST [3] tradicionales, tal y como se puede observar en el diagrama de la fig. 6, BLAST [3] ejecuta una modificación del algoritmo Smith-Waterman con el fin de alinear la secuencia objeto con todas aquellas secuencias procedentes de la base de datos que son consideradas posibles alineamientos gracias a la búsqueda realizada por BLAST [3]. Este método empleado por BLAST [3] reduce considerablemente el número de secuencias a analizar de la base de datos reduciendo así el tiempo de computación empleado por el algoritmo Smith-Waterman, pero aun así el tiempo total empleado para la ejecución de Smith-Waterman dentro de BLAST [3] abarca una gran porcentaje del tiempo total de BLAST [3], este porcentaje es dependiente de los resultados obtenidos en las fases anteriores.

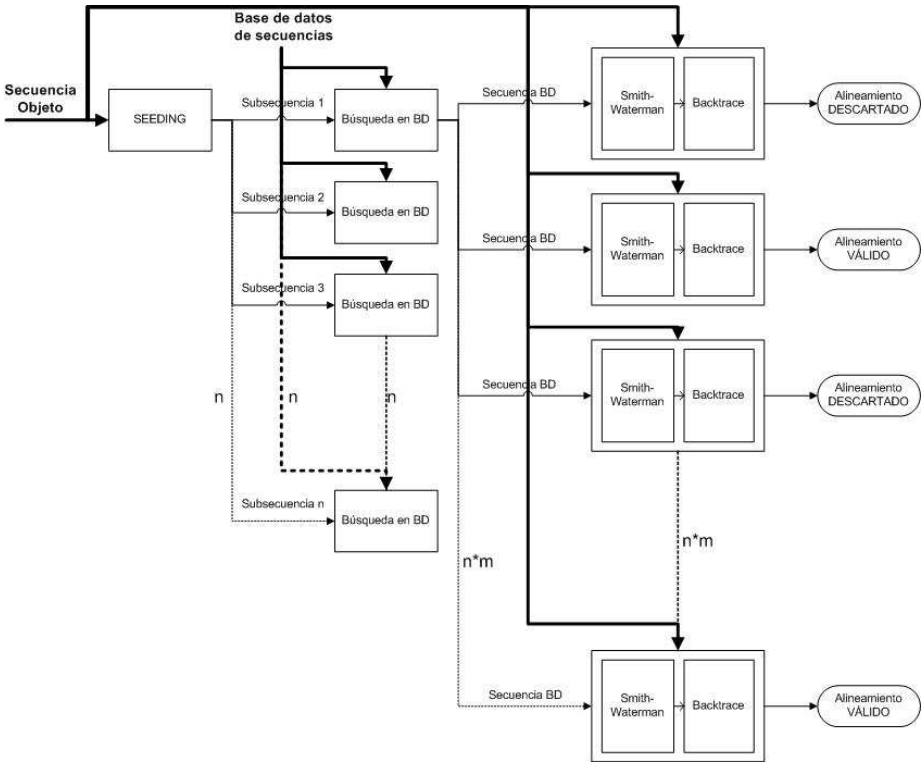


Fig. 6 Esquema de BLAST

En esta implementación del algoritmo Smith-Waterman [2], [1] para sistemas BLAST [3] se trata de realizar una poda sobre los resultados obtenidos por la búsqueda de BLAST [3], limitando de este modo el número de secuencias de base de datos que se han de procesar mediante Smith-Waterman tradicional. De este modo, todas aquellas secuencias objeto resultantes de la búsqueda que en el sistema BLAST [3] son descartadas a la hora de realizar el algoritmo Smith-Waterman, pues su valor máximo es inferior al umbral definido por el usuario, serán previamente descartadas por el Smith-Waterman

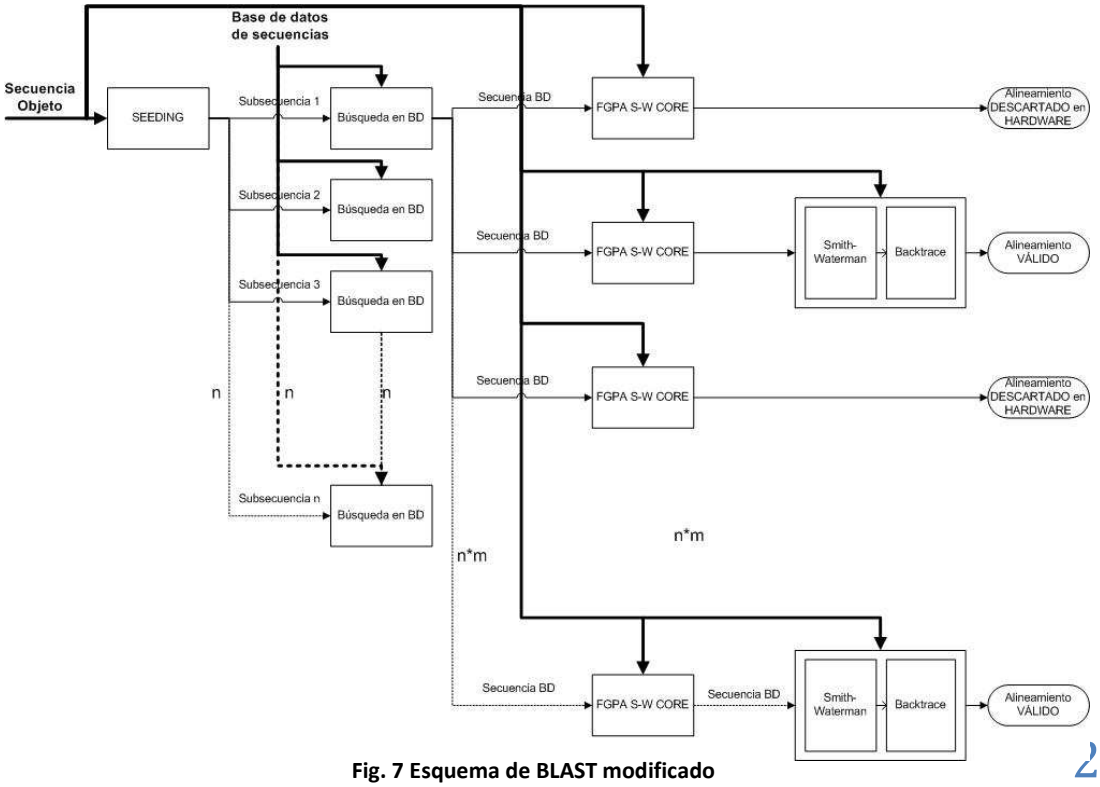


Fig. 7 Esquema de BLAST modificado

Core. Este core a nivel hardware obtiene el valor máximo del algoritmo Smith-Waterman [2], [1] entre dos secuencias a una mayor velocidad que las implementaciones software, por el contrario el procesamiento realizado no almacena la matriz de Smith-Waterman y por tanto esta ha de ser calculada completamente para aquellas secuencias que tengan que ser extendidas mediante Smith-Waterman [2], [1]. En la fig. 7 se puede observar el nuevo esquema del sistema en el cual se aprecia que aquellas secuencias descartadas en la fase final se descartan previamente mediante hardware.

4.3.Arquitectura hardware

El proyecto se encuentra enmarcado dentro del concepto codiseño hardware-software, es decir sistemas en los que la ejecución de tareas se encuentra balanceada entre ejecución software tradicional y ejecución en módulos hardware específicos. Para esta tarea se va a utilizar un sistema on-chip XD2000i fabricado por Xtreme Data [20] disponible actualmente en el laboratorio del grupo High Preformance Computing and Networking de la Universidad Autónoma de Madrid [21]. Este sistema se caracteriza por disponer de una placa instalable en uno de los zócalos de un procesador AMD Dual Core, así como de las herramientas necesarias para la comunicación de dicha placa con el sistema operativo.

La arquitectura propuesta en el sistema de codiseño se basa en la tecnología desarrollada por Impulse, el cual facilita la programación hardware mediante un lenguaje próximo a C denominado Impulse-C [22]. A nivel hardware se dispone de una FPGA Altera Startix III la cual será configurada con los cores encargados de las tareas a realizar en hardware, la comunicación entre los cores hardware y el programa software viene definida por el sistema de Impulse, y está basada en streams de datos. Este sistema de comunicación será objeto de estudio en el apartado 4.6 del presente documento.

4.4.Diseño del sistema Smith-Waterman

Debido a la complejidad presentada por el sistema BLAST [3] desarrollado por el NCBI, el presente trabajo se limita a la evaluación de secuencias biológicas en un sistema Smith-Waterman, tratando de reducir el tiempo de computación mediante la eliminación de secuencias que no superen el valor umbral dado gracias a los cores Smith-Waterman implementados.

La parte software del sistema es la encargada de obtener las secuencias a analizar, determinar en función de su tamaño si será procesada en hardware o bien si será procesada únicamente en software debido a su gran tamaño, obtener los resultados de los cores hardware y finalmente ejecutar el algoritmo Smith-Waterman [2] en aquellos alineamientos que así lo requieran. La parte hardware del sistema, la cual se estudiará más detalladamente en los apartados posteriores, es la encargada de determinar si dos secuencias dadas sobrepasan o no un valor umbral determinado.

Uno de las grandes ventajas de los sistemas hardware es la ejecución de tareas en paralelo. Mientras que en los sistemas vistos en el apartado3 el paralelismo se realizaba a la hora de analizar dos secuencias, procesando varios elementos de la matriz Smith-Waterman en paralelo, en este sistema el paralelismo se encuentra en un nivel superior realizando varias comparaciones de secuencias en paralelo, para ello es necesario

instanciar en el dispositivo FPGA diferentes cores Smith-Waterman. Dadas las restricciones de espacio establecidas por el dispositivo FGPA disponible se instanciarán tantos cores hardware como sea posible, todos con un tamaño máximo fijo de 1024 caracteres. De este modo se definirá un tamaño máximo de secuencia en el sistema y aquellas secuencias que superen el valor máximo del sistema deberán ser evaluadas completamente en software sin poder determinar previamente si superan o no el valor umbral.

El esquema global del sistema puede verse claramente esquematizado en la fig 8.

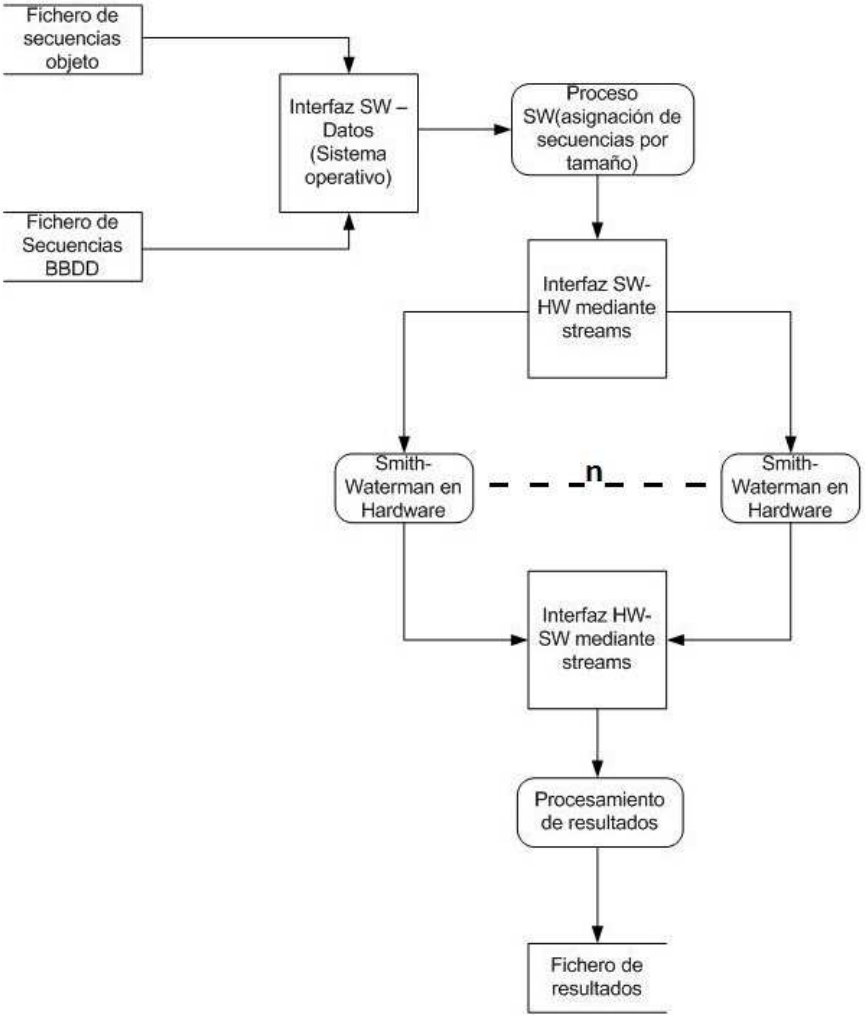


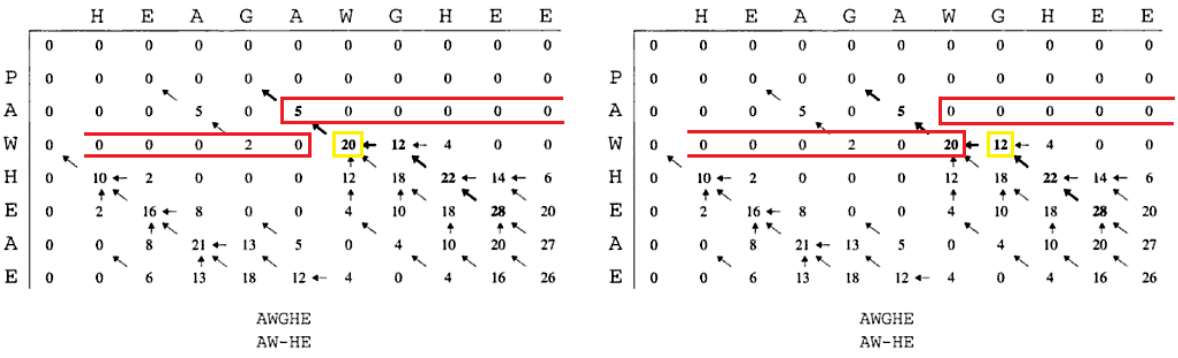
Fig. 8 Esquema del sistema completo

4.5.Diseño del Core Smith-Waterman

En este apartado se presenta el diseño del Core Smith-Waterman encargado de determinar si dos secuencias biológicas sobrepasan un valor umbral dado. Para ello el sistema necesita recibir, el valor máximo, las secuencias a procesar y la matriz de pesos correspondiente, en este apartado se va a omitir el modo en el que se obtienen los datos y se asume que el core dispone de todos ellos, posteriormente en el apartado 4.6 se determinará el modo en el que los datos son comunicados al core y la respuesta del mismo.

Para comprender el diseño implementado en este core es necesario conocer la dependencia de datos existente en el algoritmo Smith-Waterman, esta dependencia ha sido estudiada anteriormente y se muestra en la fig. 4, en la cual se puede ver que cada cela depende únicamente de la celda anterior, la celda superior y de la celda diagonal superior izquierda de la misma.

Debido a la importancia del área consumida en este tipo de sistemas, en lugar de almacenar la matriz completa necesaria para el cálculo del algoritmo, se almacenan en un registro de rotación tantos elementos como longitud tiene la secuencia procedente de la base de datos, representada generalmente en la parte superior de la matriz del algoritmo, más un elemento adicional. De este modo a la hora de calcular un elemento de la matriz se tienen almacenados los valores necesarios para su cálculo y una vez calculado, el valor de la diagonal pierde su interés ya que nunca más será utilizado. Mediante este procedimiento el nuevo valor entra al registro de rotación por la parte final y el valor de la celda diagonal, presente en el primer elemento del registro de rotación, sale del mismo quedando eliminado. Este proceso puede verse más fácilmente en fig. 9a y fig. 9b las cuales representan dos etapas del algoritmo consecutivas. En estas figuras se representa en color rojo el contenido del registro de rotación y en color amarillo el elemento a calcular, como puede observarse el valor 5 necesario para el cálculo de la celda cuyo valor es 20 sale del registro de rotación una vez calculado el valor 20 y este entra en el final del registro de rotación para el cálculo de la siguiente celda, este valor permanecerá en el registro rotacional hasta que se calculen las celdas con valor 12 y 18 presentes en la siguiente fila de la matriz.



Naterman con

Waterman con

De este modo se calcularán todos los elementos de la matriz Smith-Waterman de manera iterativa hasta encontrar un valor que supere el valor umbral o bien hasta finalizar el cálculo de la matriz y por tanto descartar el alineamiento.

En este primer diseño el tamaño de las secuencias que puede analizar el sistema está limitado por el tamaño del registro de rotación, lo cual hace inviable el sistema ya que habría que disponer de cores para cada tamaño de secuencia de base de datos. Por este motivo se propone definir segmentos de tamaños y que cada core sea capaz de procesar cualquier secuencia de base de datos cuyo tamaño sea menor que el de su registro de rotación. Para ello las secuencias de menor tamaño se rellenarán con ceros en la matriz con el fin de poder reutilizar los cores para diferentes tamaños.

4.6.Componentes hardware auxiliares

De manera adicional al core Smith-Waterman existen otros componentes hardware que completan la configuración global del sistema, estos componentes serán analizadas en este apartado con el fin de tener una visión global del sistema.

Tal y como se ha visto en el apartado 4.3 las implementaciones llevadas a cabo tienen como objetivo instanciar tantos cores Smith-Waterman como sea posible, en función de las disponibilidades de área. Por este motivo es totalmente necesario generar dos elementos capaces de obtener la información del sistema software y distribuirla entre los diferentes cores y viceversa, estos componentes se denominan distribuidor y colector.

Adicionalmente a estos sistemas y en función de la implementación, existe un último componente denominado locker cuya funcionalidad es indicar al distribuidor cual de los cores instanciados se encuentra disponible para asignarle una nueva secuencia a analizar, este componente se comporta como una cola FIFO que dispone en todo momento la información referente a los cores que han quedado liberados.

4.7.Sistema de comunicación de datos

Las comunicaciones del sistema se diferencian en tres grandes grupos, sistema de comunicación de entrada/salida del sistema, sistema de comunicación interno del sistema entre la parte software del mismo y los módulos hardware implementados y finalmente la comunicación mediante los diferentes componentes hardware.

Comunicación entrada/salida

La comunicación de entrada del sistema se realizará mediante parámetros de entrada en los cuales se definirá el valor umbral que han de superar los alineamientos para ser validos, el fichero de entrada de la base de datos, el fichero de entrada de las secuencias a analizar y finalmente la salida con los resultados obtenidos mediante el stdout. Los ficheros de entrada de base de datos y de secuencias objeto deberán mantener el formato de los ficheros recibidos por el programa BLAST del NCBI [23].

La matriz de pesos utilizada tanto en los componentes hardware como en los componentes software viene predefinida en el sistema y determina el número de bits con los que se puede codificar un elemento. En el supuesto que se desease modificar dicha matriz habría que redefinir los componentes y en función del número de elementos de dicha matriz modificar la codificación de caracteres y el empaquetamiento de los mismos.

Comunicación interna hardware/software

La comunicación entre la parte software del sistema encargada de obtener los datos y la parte hardware encargada de procesarlos se lleva a cabo mediante stream de datos de 256 bits tal y como está definida el arquitectura del sistema XD2000i utilizado.

El sistema utilizado permite el uso de stream de datos de hasta 256 bits de ancho de banda utilizando unas memorias FIFO que permiten el almacenamiento de los envíos realizados. La librería software del fabricante permite envíos de hasta 4MB de datos y siempre en un número par de bloques de 256 bits, por este motivo los últimos 4 MB puede

que no se envíen a la FPGA y se procesen únicamente en software ya que el número de bloques puede resultar impar.

El orden de envío de información al sistema hardware será modificado en función de la implementación del sistema y será explicado con detalle en los apartados específicos de cada implementación. Por el contrario el sistema de codificación será prácticamente idéntico en todas las implementaciones realizadas.

A la hora de enviar una secuencia, bien sea de base de datos u objeto, se envía previamente un bloque de 256 bits en el cual se determina la longitud de la secuencia en número de caracteres, el identificador de dicha secuencia y si es una secuencia de base de datos o una secuencia objeto. Debido al conjunto de valores disponibles para las secuencias se ha determinado la codificación de cada carácter de la secuencia en 5 bits, agrupando de este modo los 256 bits disponibles en 8 grupos de tipo int (32 bits) permite el fácil manejo de la información en sistemas software tradicionales. De estos 32 bits disponibles 30 se utilizan para codificar 6 caracteres y 2 quedan sin utilizar, por este motivo de cada 256 bits disponibles solamente se utilizan 240.

Comunicación interna de los componentes hardware

En las implementaciones realizadas existen 4 tipos de componentes diferentes y todos ellos se comunican entre sí mediante stream de datos de 256 bits. El número de stream entre componentes así como la profundidad de los mismos son elementos variables en función de la implementación utilizada.

- Distribuidor: Se compone de un stream de entrada de datos desde el exterior, un stream de salida de datos por cada core instanciado, y en determinadas implementaciones un stream de entrada desde el liberador indicando el siguiente core a procesar.
- Colector: Se compone de un stream de entrada de datos (resultados) por cada core instanciado, un stream de salida de datos al mundo exterior y en determinadas implementaciones un stream de salida con el fin de indicar al liberador que el core en cuestión queda liberado.
- Liberador: Se compone de un stream de entrada y un stream de salida, conectados con el Colector y Distribuidor respectivamente.
- Core Smith-Waterman: Se compone de un stream de entrada de datos y un stream de salida de datos conectados con el Distribuidor y el Colector respectivamente. Adicionalmente dispone de una memoria interna en la cual almacena la matriz de pesos a utilizar.

4.8.Implementación y resultados V1.0

En este apartado se va a analizar la implementación realizada en primera instancia, los resultados obtenidos, análisis de los mismos y por último posibles mejoras y posibles errores existentes.

Implementación del sistema

La primera versión del sistema se compone de 30 cores Smith-Waterman analizados en el apartado 4.5 del presente documento sin ningún tipo de mejora de

rendimiento facilitadas por la herramienta de desarrollo. Tal y como se ha indicado anteriormente estos cores disponen de una capacidad máxima de análisis de secuencias de 1024 caracteres, por lo que aquellas secuencias que superen dicho valor serán procesadas en software sin un pre-análisis que nos determine el valor máximo de las mismas.

En esta implementación la forma de operar se basa en cargar una secuencia objeto en todos los cores Smith-Waterman y posteriormente analizar contra esa secuencia objeto todas las secuencias de la base de datos. Des este modo el Distribuidor analiza si un secuencia es de tipo objeto o de tipo base de datos, en caso de que sea secuencia objeto sea envía a todos los cores para que estos realicen la carga de la misma y en caso de que sea una secuencia de base de datos localiza mediante el componente Liberador el core que se encuentra disponible y envía dicha secuencia a ese core para que sea analizada frente a la secuencia objeto enviada previamente.

Tal y como se indica en el apartado 4.7 del presente documento el último bloque de secuencias a enviar puede resultar que genere un número impar de secuencias a analizar y por tanto, debido a que en esta primera versión el sistema devuelve un bloque de 256 bits por cada secuencia analizada, no sean enviadas a la FPGA. Esto depende tanto del número total de secuencias a analizar, del tamaño de la base de datos a analizar y del número de secuencias existentes en ambos conjuntos cuyo tamaño sea mayor que 1024 caracteres.

Una vez procesados en hardware todos los elementos menores de 1024 caracteres se dispone de una matriz que indica que secuencias objeto frente a qué secuencias de base de datos superan el valor umbral dado y por tanto han de procesarse mediante software para poder obtener la matriz completa.

Ejecución y resultados

La ejecución del sistema se realiza con un conjunto de datos reducido obtenido del NCBI y simplificado en el número de secuencias objeto a analizar, debido a los elevados tiempos de computación del conjunto completo obtenido en el NCBI. El conjunto inicial estaba compuesto por 4377 secuencias objeto frente a 49999 secuencias que componen la base de datos a estudiar. El conjunto utilizado utiliza las 25 primeras secuencias objeto como subconjunto para las pruebas de rendimiento del sistema. Debido a los datos utilizados el conjunto final de 4Mb a enviar a la FPGA genera un número impar de secuencias lo que implica que estas sean procesadas únicamente en software.

En la tabla 1 se especifican los resultados obtenidos en las diferentes ejecuciones, cada tipo de ejecución viene determinada por una fila de datos y le corresponden tres ejecuciones reales con el fin de evitar posibles alteraciones externas del sistema. Los diferentes tipos de ejecuciones vienen determinados por una ejecución de referencia puramente software (SW) y una serie de ejecuciones con el sistema hardware-software con la variación del valor umbral del sistema.

El valor umbral es un valor clave en las ejecuciones del sistema ya que en función de este valor el número de secuencias descartadas en software será mayor o menor determinando el tiempo de computo software, pero del mismo modo cuanto mayor sea este valor umbral mayor será el tiempo de computo hardware ya que este deberá realizar un mayor número de iteraciones para poder llegar a un valor máximo más elevado.

Tal y como se puede comprobar en la tabla 1 los resultados obtenidos distan mucho de los resultados deseados, ya que el tiempo empleado en hardware es claramente superior al tiempo empleado en software en procesar las secuencias descartadas, de este modo se produce una desaceleración del software mayor cuanto mayor es el valor umbral.

	Tiempo total		Acelaración	Tiempo total		Acelaración	Tiempo total		Acelaración	Media
	Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		
	Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		
SW	**3695**		**1**	**3724**		**1**	**3741**		**1**	3720
	0	3695	--	0	3724	--	0	3741	--	
	1249975	0		1249975	0		1249975	0		
HW 15	**4181**		**0,8897**	**4139**		**0,8988**	**4137**		**0,8992**	4152,33
	466	3715	--	466	3673	--	466	3671	--	
	1249248	727		1249248	727		1249248	727		
HW 30	**5730**		**0,6492**	**5722**		**0,6501**	**5731**		**0,6491**	5727,67
	2043	3687	--	2043	3679	--	2043	3688	--	
	1208676	41299		1208676	41299		1208676	41299		
HW 45	**7848**		**0,4740**	**7846**		**0,4741**	**7848**		**0,4740**	7847,33
	4227	3621	--	4227	3619	--	4227	3621	--	
	1140025	109950		1140025	109950		1140025	109950		
HW 60	**10163**		**0,3660**	**10160**		**0,3661**	**10158**		**0,3662**	10160,33
	6630	3533	--	6630	3530	--	6630	3528	--	
	1003391	256584		1003391	256584		1003391	256584		

Tiempo FPGA

Tiempo Software

Núm. Secuencias procesadas en Software

Núm. Secuencias omitidas en Software

Tabla1. Resultados V1.0

Análisis del sistema

Una vez vistos los primeros resultados del sistema en los cuales se obtiene un rendimiento muy inferior al esperado, se realiza un profundo análisis del sistema mediante simulaciones con datos reales hasta encontrar el origen de los problemas. Tal y como se puede observar en la Fig.10 durante una gran cantidad de ciclos el sistema se mantiene pasando por un conjunto de estados los cuales representan la rotación del registro de forma secuencial.

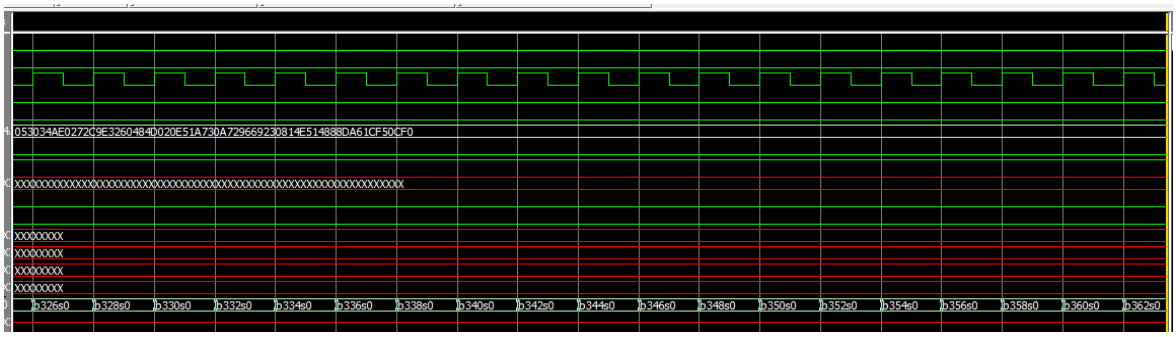


Fig. 10 Simulación Versión 1.0

Inicialmente se pensó en albergar la información necesaria para la siguiente ejecución de la matriz de pesos de Smith-Waterman en un registro de rotación, de tal forma que el coste de rotar dicho registro fuese mínimo. Debido a las características del sistema Impulse-C [22] la implementación de este registro de rotación se realizó mediante BRAMs por lo que el coste de rotar dicho 'registro' en cada elemento de la matriz supone un coste de computación excesivamente elevado, haciendo que cuanto mayor fuese el número de elementos de la matriz a calcular mucho mayor fuese el coste hardware de ejecución del algoritmo. Este incremento del coste viene dado porque en cada ciclo de reloj solamente se puede acceder a dos posiciones de memoria en el mejor de los casos, lo que implica que para rotar una array de n elementos se requieren mínimo n ciclos de reloj, n/2 de lectura y n/2 de escritura en la BRAM disponible.

Mejoras del sistema

Tal y como se ha comentado en el punto anterior, la versión de la herramienta de desarrollo utilizada para el desarrollo de la primera versión del sistema no permite el uso de registros para almacenaje de grandes bloques de información, esto implica que el registro en el cual se almacena la información de la matriz procesada y necesaria para el cálculo de la próxima celda se encuentra en memoria, y que por tanto la rotación del mismo exige una gran cantidad de ciclos de reloj puesto que no se pueden escribir y leer múltiples posiciones de memoria de manera simultánea dentro de una misma Block-RAM. En la última versión de Impulse-C [22] se permite el uso de registro para almacenar información de forma específica lo cual indica que una implementación con estas características deberá reducir el tiempo de ejecución de cada core, por el contrario el consumo de área de cada core se verá incrementado y por tanto el número de elementos de procesamiento en paralelo deberá ser menor. Tras la implementación del sistema en su versión 1.1 utilizando los registros disponibles en la versión 3.7 de Impulse-C se ve como el consumo de área se incrementa de manera elevada y descarta dicha opción por el reducido número de cores disponibles.

Otra de las carencias que se presenta la propuesta inicial del sistema es el envío repetido de información desde la interfaz software al sistema hardware ya que la base de datos se envía completa tantas veces como secuencias objeto se quieran analizar. La solución de este problema queda descartada para la siguiente implementación ya que el incremento del tiempo de ejecución en FPGA del sistema en función del valor umbral hace ver que la mayor parte consumo no es en el envío de información si no en el procesamiento de la misma puesto que el envío de datos es el mismo para los cuatro tipos de ejecuciones realizadas.

Respecto al envío de información existe otra carencia comentada anteriormente y es el envío de un número de secuencias impar al sistema hardware, ya que produce la

recepción de un número impar de bloques de 256 bits y por tanto el bloqueo del sistema, debido a las restricciones existentes en la plataforma utilizada.

4.9.Implementación y resultados V2.0

En este apartado se van a analizar las modificaciones presentadas por el sistema respecto a la versión 1.0 estudiada en el apartado 4.8, los resultados obtenidos en la nueva versión, análisis de los mismos y por último posibles mejoras y posibles errores existentes.

Modificaciones del sistema

La versión 2.0 del sistema trata de solucionar el problema existente en la rotación del registro debido a que dicho registro se encuentra almacenado en un BRAM. Para ello en lugar de tratar de utilizar un registro como en la versión 1.1, lo cual implica un alto coste de área, se mantiene la implementación mediante BRAM pero se realiza una lista circular de forma virtual manipulando los índices de acceso a la BRAM por lo que el coste de rotación queda reducido a incrementar un índice y comprobar si este ha sobrepasado el máximo para reiniciarlo. Con esta modificación se espera una mejora sustancial en la parte hardware del sistema, y de este modo poder comprobar si se obtiene una mejora de carácter global.

El sistema se mantiene sin la utilización de mejoras disponibles en la herramienta de desarrollo como son los pragmas de configuración hardware.

Recursos utilizados por el sistema hardware

A continuación se muestra un resumen de los recursos consumidos por el sistema en el dispositivo FPGA utilizado, tal y como se puede observar, los recursos no se encuentran utilizados al máximo de sus capacidades, por el contrario no es posible añadir más cores de procesamiento debido a que los bloques M9K disponibles en el sistema se están usando un total de 864 / 864 (100 %).

Fitter Status	Successful - Tue Aug 16 18:57:27 2011
Quartus II Version	9.0 Build 132 02/25/2009 SJ Full Version
Revision Name	appA_top
Top-level Entity Name	app_top
Family	Stratix III
Device	EP3SE260F1152C3
Timing Models	Final
Logic utilization	68%
Combinational ALUTs	83,905 / 203,520 (41 %)
Memory ALUTs	144 / 101,760 (< 1 %)
Dedicated logic registers	113,315 / 203,520 (56 %)
Total registers	114106
Total pins	436 / 744 (59 %)
Total virtual pins	0
Total block memory bits	3,163,552 / 15,040,512 (21 %)

```
DSP block 18-bit elements      0 / 768 ( 0 % )
Total PLLs                     4 / 8 ( 50 % )
Total DLLs                     1 / 4 ( 25 % )
+------------------------------+----------------------------------------+
```

Ejecución y resultados

De cara a mantener una uniformidad en las pruebas realizadas, los datos utilizados en estas pruebas se mantienen constantes respecto a las pruebas de la versión 1.0. Del mismo modo se mantiene la carencia del sistema del número de secuencias impar en el último bloque de secuencias enviadas a la FPGA.

En la tabla 2 se especifican los resultados obtenidos en las diferentes ejecuciones, cada tipo de ejecución viene determinada por una fila de datos y le corresponden tres ejecuciones reales con el fin de evitar posibles alteraciones externas del sistema. Los diferentes tipos de ejecuciones vienen determinados por una ejecución de referencia puramente software (SW) y una serie de ejecuciones utilizando el sistema hardware-software con la variación del valor umbral del sistema, el cual es determinante en el coste de ejecución tanto de la parte hardware como del número de secuencias omitidas en la parte software.

	Tiempo total		Acelar ación	Tiempo total		Acelar ación	Tiempo total		Acelar ación	Media
	Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		
	Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		
SW	**3695**		**1**	**3689**		**1**	**3702**		**1**	3695,33
	0	3695	--	0	3689	--	0	3702	--	
	1249975	0		1249975	0		1249975	0		
HW 0 Carga Datos	**3688**		**1,0020**	**3683**		**1,0033**	**3685**		**1,0028**	3685,33
	8	3680	--	8	3675	--	8	3677	--	
	1249975	0		1249975	0		1249975	0		
HW 30	**3762**		**0,9823**	**3725**		**0,9920**	**3728**		**0,9912**	3738,33
	54	3708	--	54	3671	--	54	3674	--	
	1208676	41299		1208676	41299		1208676	41299		
HW 60	**3719**		**0,9936**	**3719**		**0,9936**	**3720**		**0,9934**	3719,33
	161	3558	--	161	3558	--	161	3559	--	
	1003391	256584		1003391	256584		1003391	256584		
HW 90	**3556**		**1,0392**	**3555**		**1,0395**	**3554**		**1,0398**	3555,00
	276	3280	--	276	3279	--	276	3278	--	
	767489	482486		767489	482486		767489	482486		
HW	**3230**		**1,1441**	**3232**		**1,1434**	**3230**		**1,1441**	3230,67

120	380	2850	--	380	2852	--	380	2850	--	
	560223	689752		560223	689752		560223	689752		
HW 150	**2785**		**1,3269**	**2784**		**1,3273**	**2785**		**1,3269**	2784,67
	462	2323	--	462	2322	--	462	2323	--	
	383307	866668		383307	866668		383307	866668		
HW 180	**2293**		**1,6116**	**2291**		**1,6130**	**2292**		**1,6123**	2292,00
	522	1771	--	522	1769	--	522	1770	--	
	245025	1004950		245025	1004950		245025	1004950		
HW 210	**1803**		**2,0495**	**1804**		**2,0484**	**1804**		**2,0484**	1803,67
	562	1241	--	562	1242	--	562	1242	--	
	144634	1105341		144634	1105341		144634	1105341		
HW 240	**1445**		**2,5573**	**1445**		**2,5573**	**1445**		**2,5573**	1445,00
	586	859	--	586	859	--	586	859	--	
	87755	1162220		87755	1162220		87755	1162220		
HW 270	**1240**		**2,9801**	**1240**		**2,9801**	**1240**		**2,9801**	1240,00
	599	641	--	599	641	--	599	641	--	
	61440	1188535		87755	1162220		87755	1162220		
HW 300	**1126**		**3,2818**	**1123**		**3,2906**	**1126**		**3,2818**	1125,00
	607	519	--	607	516	--	607	519	--	
	48660	1201315		48660	1201315		48660	1201315		
HW 330	**1056**		**3,4994**	**1056**		**3,4994**	**1056**		**3,4994**	1056,00
	612	444	--	612	444	--	612	444	--	
	42518	1207457		42518	1207457		42518	1207457		
HW 360	**1009**		**3,6624**	**1009**		**3,6624**	**1010**		**3,6587**	1009,33
	615	393	--	615	394	--	615	395	--	
	39036	1210939		39036	1210939		39036	1210939		
HW 390	**971**		**3,8057**	**971**		**3,8057**	**971**		**3,8057**	971,00
	616	355	--	616	355	--	616	355	--	
	36683	1213292		36683	1213292		36683	1213292		
HW 420	**948**		**3,8980**	**948**		**3,8980**	**949**		**3,8939**	948,33
	617	331	--	617	331	--	617	332	--	
	35381	1214594		35381	1214594		35381	1214594		
HW 450	**931**		**3,9692**	**931**		**3,9692**	**931**		**3,9692**	931,00
	618	313	--	618	313	--	618	313	--	
	34561	1215414		34561	1215414		34561	1215414		
HW 480	**919**		**4,0210**	**919**		**4,0210**	**919**		**4,0210**	919,00
	618	301	--	618	301	--	6.189	-5270	--	
	34072	1215903		34072	1215903		34072	1215903		

HW 510	**915**		4,0386	--	**915**		4,0386	--	**915**		4,0386	--	915,00
	618	297			618	297			618	297		
	33880	1216095			33880	1216095			33880	1216095		
HW 540	**913**		4,0475	--	**913**		4,0475	--	**913**		4,0475	--	913,00
	618	295			618	295			618	295		
	33812	1216163			33812	1216163			33812	1216163		
HW 570	**913**		4,0475	--	**913**		4,0475	--	**913**		4,0475	--	913,00
	618	295			618	295			618	295		
	33806	1216169			33806	1216169			33806	1216169		

Tiempo FPGA

Tiempo Software

Núm. Secuencias procesadas en Software

Núm. Secuencias omitidas en Software

Tabla2. Resultados V2.0

Análisis del sistema

Una vez vistos los resultados del mostrados en la tabla 2, los resultados obtenidos en la versión 2.0 mejoran de forma considerable los obtenidos en la primera versión del sistema. Desde este momento se puede empezar a realizar una valoración del sistema de carácter global ya que por primera vez se ve la relación entre el valor umbral indicado y la aceleración obtenida. En esta versión del sistema de codiseño hardware–software se obtiene una aceleración del sistema que varía entre 1x para valores de umbral pequeños hasta una aceleración ligeramente superior a 4x para valores de umbral máximos. Es muy importante destacar que la aceleración obtenida es una aceleración completamente dependiente tanto del conjunto de datos utilizado como del valor umbral requerido por lo que es complicado determinar una aceleración global del sistema.

En el apartado 4.8 se analiza como una posible mejora el modo de envío de información de tal modo que no se reenvíe la información repetida de forma innecesaria. Esta mejora no está implementada en la versión 2.0 del sistema, pero tal y como se ve en Tabla 2 fila HW 0 el coste de enviar toda la información a la FPGA es de tan solo 8 segundos, lo que representa un 0,2% del tiempo total de la ejecución de referencia software.

Es muy importante analizar los valores obtenidos en HW 120, esta ejecución es la primera ejecución en la cual se omiten más pares de secuencias omitidas de las que se ejecutan y aun así la mejora del sistema para este valor es de tan solo 1,14x. Esto se debe a que el gran consumo de tiempo en la ejecución del sistema se debe a las grandes secuencias a alinear, las cuales generan una gran matriz de resultados y por norma general un elevado valor máximo en dicha matriz.

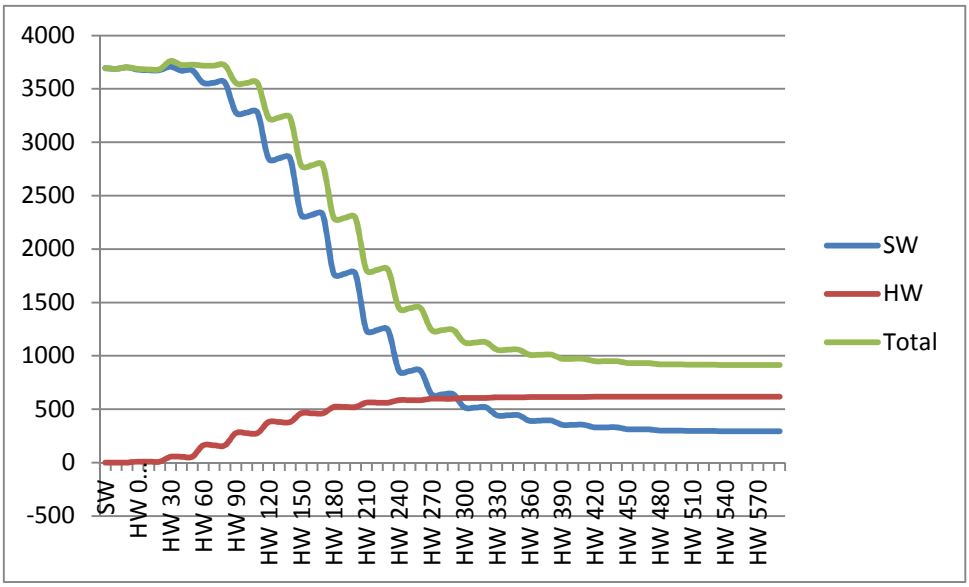


Gráfico 1 Análisis de resultados v2.0

En el gráfico 1 se muestra una comparativa de forma evolutiva de los resultados obtenidos en la tabla 2, tal y como se puede observar, inicialmente el 100% del tiempo de ejecución era utilizado por la parte software, y según se aumenta el valor umbral deseado el tiempo comienza a ser empleado por la parte hardware.

Mejoras del sistema

Se mantienen las mejoras pendientes de la versión 1.0 en lo referente al envío de información duplicada al sistema y a la posibilidad de analizar un número impar de secuencias.

Hay que valorar las posibles modificaciones a nivel hardware presentadas por la herramienta de desarrollo como por ejemplo la creación de elementos pipeline.

Otra posible alternativa es la creación de un sistema con un mayor tamaño máximo de cadena de este modo se podrán descartar muchas más secuencias y con mayor coste de computación desde la fase hardware.

4.10. Implementación y resultados V3.0

En este apartado se van a analizar las modificaciones presentadas por el sistema respecto a la versión 2.0 estudiada en el apartado 4.9, los resultados obtenidos en la nueva versión, análisis de los mismos y por último posibles mejoras y errores existentes.

Modificaciones del sistema

Se han producido tres grandes modificaciones respecto a la versión anterior del sistema, a continuación se detalla cada una de las modificaciones realizadas y la repercusión de la misma sobre el sistema.

- **Pipeline.** En primer lugar se ha realizado un pipeline al nivel más profundo del algoritmo, de este modo el proceso de cálculo de cada celda se ha visto claramente

acelerado gracias al uso de esta técnica. En Impulse-C [22], tecnología utilizada para este proyecto, el pipeline se define mediante una sentencia #pragma co pipeline al inicio de un bucle y automáticamente la transformación del código de alto nivel a vhdl realiza el pipeline. Inicialmente se ha generado un pipeline de rate 6, aunque modificando las sentencias empleadas y la forma de acceder a los recursos, este se ha podido bajar hasta rate 2. Finalmente la versión utilizada ha sido el pipeline con rate 3 ya que el que obtenía inicialmente un rate 2 generaba en la simulación en modelsim, y posteriormente en el dispositivo hardware, resultados erróneos al analizar secuencias con resultado negativo como resultado positivo, error producido por la lectura de datos desde las BRAM.

Esta mejora es una mejora de rendimiento que afecta a todas las secuencias procesadas en hardware ya que únicamente afecta al cálculo de la matriz en la versión hardware.

- **Tamaño máximo admitido.** Se ha ampliado el tamaño máximo de secuencia admitida hasta 2047 elementos, de este modo se permite analizar en hardware una mayor cantidad de secuencias que antes únicamente se procesaban en software. El motivo de usar 2047 es que el array debe de ser de un elemento más que la secuencia y utilizando este valor se produce una mejora en el consumo de memoria del sistema hardware en la FPGA.

Esta mejora produce una mayor cantidad de secuencias a analizar en hardware y por tanto la posibilidad de descartar dichas secuencias reduciendo el tiempo de computación software. Es una mejora dependiente del conjunto de datos a emplear.

- **Aceptación de un número impar de secuencias.** A pesar de parecer la menos importante de las mejoras realizadas en esta versión, resulta ser una de las que mayor importancia tiene a la hora del rendimiento. Una de las restricciones que presenta el sistema de codiseño utilizado es que siempre se deben enviar y recibir un número par de bloques, en la primera versión del sistema cada par de sentencias analizadas devolvía un solo bloque haciendo que no se pudiesen mandar a analizar un número impar de secuencias y por tanto condenando al sistema software a procesar el último bloque de 4 MB de secuencias en caso de que este albergase un número impar de elementos.

En la versión 3.0 cada par analizado devuelve un bloque con el resultado y un bloque en blanco reduciendo ligeramente el rendimiento en el envío de información pero permitiendo un número de secuencias impares. Esta mejora de rendimiento depende del conjunto de datos utilizado así como del tamaño máximo admitido en el sistema, ya que en función de estas dos variables el último bloque albergará un número par de secuencias o no. En el caso de los datos analizados el número de secuencias incluidas en el último bloque es de 37000 secuencias, que en la versión anterior no eran analizadas previamente en hardware.

Recursos utilizados por el sistema hardware

A continuación se muestra un resumen de los recursos consumidos por el sistema en el dispositivo FPGA utilizado, tal y como se puede observar los recursos no se encuentran utilizados al máximo de sus capacidades, por el contrario no es posible añadir más cores de procesamiento debido a que los bloques M9K disponibles en el sistema se están usando un total de 864 sobre 864 disponibles.

+---+

Fitter Status	Successful - Tue Aug 16 18:57:27 2011
Quartus II Version	9.0 Build 132 02/25/2009 SJ Full Version

Revision Name	appA_top
Top-level Entity Name	app_top
Family	Stratix III
Device	EP3SE260F1152C3
Timing Models	Final
Logic utilization	68%
Combinational ALUTs	84,213 / 203,520 (41 %)
Memory ALUTs	144 / 101,760 (< 1 %)
Dedicated logic registers	114,952 / 203,520 (56 %)
Total registers	114106
Total pins	436 / 744 (59 %)
Total virtual pins	0
Total block memory bits	3,365,495 / 15,040,512 (22 %)
DSP block 18-bit elements	0 / 768 (0 %)
Total PLLs	4 / 8 (50 %)
Total DLLs	1 / 4 (25 %)

+------------------------------+--+

Ejecución y resultados

De cara a mantener una uniformidad en las pruebas realizadas, los datos utilizados en estas pruebas se mantienen constantes respecto a las pruebas realizadas en las versiones anteriores del sistema.

En la tabla 3 se especifican los resultados obtenidos en las diferentes ejecuciones, cada tipo de ejecución viene determinada por una fila de datos y le corresponden tres ejecuciones reales con el fin de evitar posibles alteraciones externas del sistema. Los diferentes conjuntos de ejecuciones vienen determinados por una ejecución de referencia puramente software (SW) y una serie de ejecuciones utilizando el sistema hardware-software con la variación del valor umbral del sistema, el cual es determinante en el coste de ejecución tanto de la parte hardware como del número de secuencias omitidas en la parte software.

	Tiempo total		Acelaración	Tiempo total		Acelaración	Tiempo total		Acelaración	Media
	Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		Tiemp fpga	Tiempo Sw		
	Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		Sec SW	Sec Omitidas		
SW	**3695**		**1**	**3689**		**1**	**3702**		**1**	3695,33333
	0	3695	--	0	3689	--	0	3702	--	
	1249975	0		1249975	0		1249975	0		
HW 0 Carga Datos	**3706**		**0,9971**	**3704**		**0,9977**	**3703**		**0,9979**	3704,33
	9	3697	--	9	3695	--	9	3694	--	
	1249975	0		1249975	0		1249975	0		

HW 30	3737		0,9889	3739		0,9883	3736		0,9891	3737,33
	14	3723	--	14	3725	--	14	3722	--	
	1208434	41541		1208434	41541		1208434	41541		
HW 60	3602		1,0259	3602		1,0259	3604		1,0253	3602,67
	30	3572	--	30	3572	--	30	3574	--	
	1001392	248583		1001392	248583		1001392	248583		
HW 90	3339		1,1067	3340		1,1064	3340		1,1064	3339,67
	47	3292	--	47	3293	--	47	3293	--	
	762091	487884		762091	487884		762091	487884		
HW 120	2900		1,2743	2900		1,2743	2901		1,2738	2900,33
	62	2838	--	62	2838	--	62	2839	--	
	551349	698626		551349	698626		551349	698626		
HW 150	2361		1,5652	2361		1,5652	2361		1,5652	2361,00
	74	2287	--	74	2287	--	74	2287	--	
	370688	879287		370688	879287		370688	879287		
HW 180	1784		2,0714	1783		2,0725	1783		2,0725	1783,33
	83	1.701	--	83	1700	--	83	1700	--	
	227456	1022519		227456	1022519		227456	1022519		
HW 210	1214		3,0439	1214		3,0439	1214		3,0439	1214,00
	89	1125	--	89	1125	--	89	1125	--	
	121261	1128714		121261	1128714		121261	1128714		
HW 240	802		4,6076	802		4,6076	802		4,6076	802,00
	92	710	--	92	710	--	92	710	--	
	59773	1190202		59773	1190202		59773	1190202		
HW 270	543		6,8054	543		6,8054	543		6,8054	543,00
	94	449	--	94	449	--	94	449	--	
	31621	1218354		31621	1218354		31621	1218354		
HW 300	401		9,2153	401		9,2153	401		9,2153	401,00
	95	306	--	95	306	--	95	306	--	
	17906	1232069		17906	1232069		17906	1232069		
HW 330	324		11,4053	324		11,4053	324		11,4053	324,00
	96	228	--	96	228	--	96	228	--	
	11429	1238546		11429	1238546		11429	1238546		
HW 360	271		13,6359	271		13,6359	271		13,6359	271,00
	97	174	--	97	174	--	97	174	--	
	7809	1242166		7809	1242166		7809	1242166		
HW 390	225		16,4237	225		16,4237	225		16,4237	225,00

	97	128	--	97	128	--	97	128	--	
	5197	1244778		5197	1244778		5197	1244778		
HW 420	**194**		19,0481	**194**		19,0481	**194**		19,0481	194,00
	97	97	--	97	97	--	97	97	--	
	3641	1246334		3641	1246334		3641	1246334		
HW 450	**170**		21,7373	**170**		21,7373	**170**		21,7373	170,00
	97	73	--	97	73	--	97	73	--	
	2603	1247372		2603	1247372		2603	1247372		
HW 480	**151**		24,4724	**151**		24,4724	**151**		24,4724	151,00
	97	54	--	97	54	--	97	54	--	
	1888	1248087		1888	1248087		1888	1248087		
HW 510	**137**		26,9732	**137**		26,9732	**137**		26,9732	137,00
	97	40	--	97	40	--	97	40	--	
	1439	1248536		1439	1248536		1439	1248536		
HW 540	**131**		28,2087	**131**		28,2087	**131**		28,2087	131,00
	97	34	--	97	34	--	97	34	--	
	1251	1248724		1251	1248724		1251	1248724		
HW 570	**130**		28,4256	**130**		28,4256	**130**		28,4256	130,00
	97	33	--	97	33	--	97	33	--	
	1221	1248754		1221	1248754		1221	1248754		
HW 5000	**129**		28,7158	**129**		28,7158	**129**		28,7158	129,00
	97	32	--	97	32	--	97	32	--	
	1100	1248875		1100	1248875		1100	1248875		

Tiempo FPGA	Núm. Secuencias procesadas en Software	Núm. Secuencias omitidas en Software
Tiempo Software		

Tabla3. Resultados v3.0

Análisis del sistema

Una vez vistos los resultados presentes en la tabla 3 se puede comprobar una gran mejora en el sistema respecto a la versión anterior. En la versión 3.0 se obtiene una aceleración máxima superior a 28x mientras que en la versión anterior el valor máximo obtenido era de 5x. Del mismo modo que en la versión anterior la penalización del sistema

para el caso peor es prácticamente nula, ralentizando el sistema hasta 0,9889x en el peor de los casos probados.

Cuanto mayor el valor umbral del sistema mayor es el número de pares de secuencias que no llegan a dicho valor y por tanto menor carga de trabajo para la parte software. Así mismo la parte hardware se ve incrementada en tiempo de computo ya que son más las matrices Smith-Waterman que hay completar para poder asegurar que no superan el valor dado.

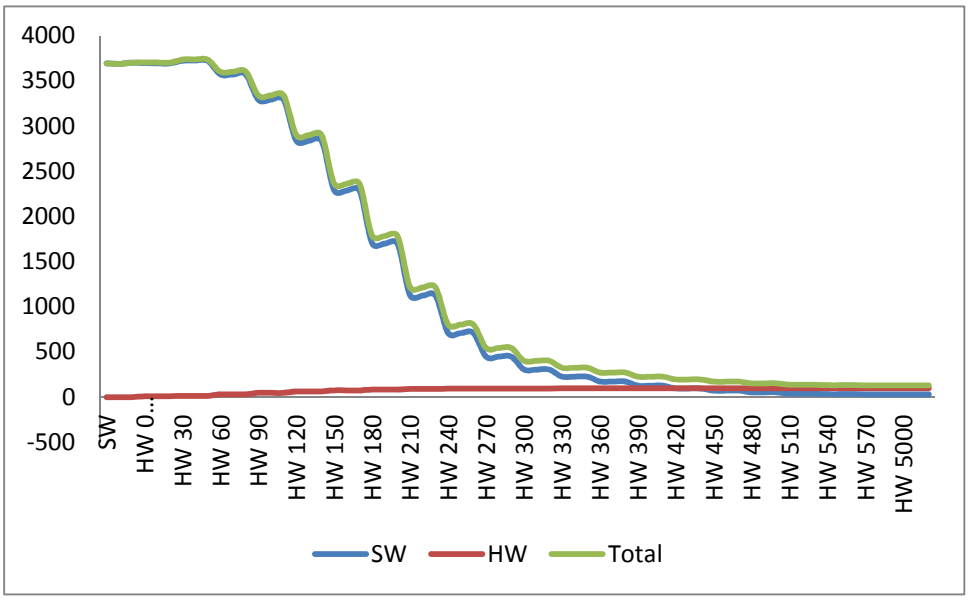


Gráfico 2 Análisis de resultados v3.0

Tal y como se puede comprobar en el gráfico 2 la gran cantidad de tiempo consumido por el sistema se debe a la parte software del mismo, siendo está más elevada cuanto menor es el valor umbral seleccionado. Por el contrario para valores umbral superiores a 300 la mejora del sistema debido a los descartes realizados en hardware hace que el coste de procesar el resto de secuencias en software se reduzca en gran medida respecto a su coste original.

Al igual que en la versión 2.0, es muy importante analizar los valores obtenidos en HW 120 debido a que esta ejecución es la primera ejecución en la cual se omiten más pares de secuencias de las que se procesan finalmente y aun así la mejora del sistema para este valor es de tan solo 1,27x. Esto se debe a que el gran consumo de tiempo en la ejecución del sistema es causado por las grandes secuencias a alinear, las cuales generan una gran matriz de resultados y por norma general un elevado valor máximo en dicha matriz.

5. CONCLUSIONES Y TRABAJOS FUTUROS

La gran evolución sufrida por las bases de datos de secuencias genéticas utilizadas por la comunidad científica durante las últimas décadas, ha hecho que los algoritmos de alineación de secuencias biológicas utilizados se hayan quedado obsoletos. Por este motivo, durante los años 90, surgieron algoritmos heurísticos como BLAST [3] que solventaron el problema de forma temporal tras ganarse la confianza de la comunidad científica.

Hoy en día, dos décadas después, nos encontramos con el mismo problema pero a mayor escala. Los algoritmos heurísticos utilizados, como BLAST [3] y BLAST2 [7], se han vuelto a quedar obsoletos produciendo unos tiempos de ejecución demasiado elevados para su uso de forma cotidiana frente a las grandes bases de datos existentes. Por este motivo es necesario evolucionar la tecnología de alineación de secuencias genéticas al siguiente escalón disponible, la aceleración de algoritmos mediante hardware.

En este trabajo de fin de máster se ha profundizado en el algoritmo BLAST [3] y más concretamente en un algoritmo utilizado por este denominado Smith-Waterman [2], dicho algoritmo consume una gran parte del tiempo de ejecución de BLAST [3] y este ha sido el motivo de que se tratase como parte central del trabajo de fin de máster. Finalmente se ha conseguido realizar una variante válida para BLAST [3] del algoritmo Smith-Waterman [2] mediante un sistema HPRC. Esta modificación tiene la salvedad de que solamente calcula las matrices Smith-Waterman [2] para aquellos alineamientos cuyo valor máximo supere un determinado valor umbral, lo cual implica que estos serán los utilizados por BLAST [3] en sus fases posteriores y por tanto la salida de nuestra modificación de Smith-Waterman [2] es una entrada perfectamente válida para la siguiente fase de BLAST [3].

En esta modificación del algoritmo de Smith-Waterman [2] se ha obtenido una aceleración de hasta 28 veces mejor que el rendimiento de la versión software en el mismo sistema. Para ello se ha utilizado un subconjunto de datos obtenidos del NCBI [8], tanto como base de datos como conjunto de secuencias objeto. El rendimiento obtenido ha sido mayor cuanto más se ha aumentado el valor umbral puesto que mayor número de secuencias son descartadas a la hora de ejecutar la versión software del alineamiento, esto hace que el sistema actual tenga un rendimiento totalmente dependiente de los datos de entrada.

Otro de los factores implicados en el rendimiento del sistema, a parte de la implementación diseñada, es el tipo de arquitectura HPRC utilizada. En este caso se ha utilizado un acelerador XD2000i [20] de Xtreme Data la cual utiliza tecnología in-socket. Esta tecnología ofrece una gran velocidad de transmisión de datos desde el sistema software en detrimento de utilizar más elementos de cálculo. En este caso debido al paralelismo existente en el algoritmo, ya que el número de secuencias a procesar en paralelo es considerablemente grande, el sistema encajaría mejor en una arquitectura de tarjetas en bus de datos (por ejemplo PCIe) la cual ralentiza el envío de información pero aumenta el área disponible para computación en paralelo. Esta afirmación concuerda con los resultados obtenidos en el trabajo presentado en [19] donde se ha desarrollado un sistema similar al del presente documento pero con arquitectura de tarjetas en el bus PCI, obteniendo en determinados casos muy concretos un rendimiento de hasta 750x.

5.1.Trabajo futuro.

Por último, las líneas de investigación futuras van dirigidas en tres grandes puntos mediantes los cuales se puede obtener una implementación realmente interesante de cara a implementar un sistema BLAST [3] completo.

1. Optimización a nivel arquitectura Hardware. En este primer análisis de viabilidad se han obtenido importantes aceleraciones partiendo desde una descripción Impulse-C. Optimizaciones arquitecturales permitieron acelerar desde 4x a 28x. No obstante análisis más profundos del hardware sintetizado desde alto nivel daría ahorros de área y aumento de velocidad.

2. Arquitectura escalable. Tal y como se ha comentado en el punto anterior del presente documento, este algoritmo es más acorde a arquitecturas escalables mediante tarjetas aceleradores en el bus de datos, por ejemplo tarjetas PCIe. Obteniendo de este modo una gran capacidad de procesamiento en paralelo. Incluso la alternativa de múltiples aceleradores funcionando en paralelo es perfectamente viable con la arquitectura utilizada.

3. Tamaño máximo de secuencia. Actualmente el tamaño máximo de secuencia admitido por la parte hardware del sistema queda limitado a 2047 caracteres por secuencias, siendo este igual para todos los cores implementados. Con el fin de no perder área de forma innecesaria sería interesante desarrollar cores de diferentes tamaños máximos así como un sistema de repartición de secuencias en función de tamaño y disponibilidad de los cores, y no solo por disponibilidad tal y como se realiza actualmente.

4. Conjunto de datos variable. El sistema actual está diseñado para el análisis de secuencias de aminoácidos. En trabajos futuros se debe permitir la posibilidad de cargar la matriz de pesos de forma manual, así como la codificación utilizada por el sistema a la hora de enviar los caracteres de las secuencias.

6. REFERENCIAS

[1] T.F. Smith and M.S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, no. 4, pp. 482-489, 1981.

[2] TF Smith and MS Waterman, "Identification of common molecular subsequences," *J. Mol. Bwl*, vol. 147, pp. 195-197, 1981.

[3] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman, "Basic local alignment search tool," *Journal of molecular biology*, vol. 215, no. 3, pp. 403-410, 1990.

[4] H.Y. Liao, M.L. Yin, and Y. Cheng, "A parallel implementation of the Smith-Waterman algorithm for massive sequences searching," in *IEEE*, vol. 2, 2005, pp. 2817-2820.

[5] B. Harris, A.C. Jacob, J.M. Lancaster, J. Buhler, and R.D. Chamberlain, "A banded Smith-Waterman FPGA accelerator for Mercury BLASTP," in *IEEE*, 2007, pp. 765-769.

[6] Wikipedia. Sequence alignment. [Online]. http://en.wikipedia.org/wiki/Sequence_alignment

[7] T.A. Tatusova and T.L. Madden, "BLAST 2 Sequences, a new tool for comparing protein and nucleotide sequences," *FEMS microbiology letters*, vol. 174, no. 2, pp. 247-250, 1999.

[8] US government-funded national resource for molecular biology information. National Center for Biotechnology Information. [Online]. http://www.ncbi.nlm.nih.gov/

[9] BLAST Wikipedia. [Online]. http://en.wikipedia.org/wiki/BLAST

[10] S.B. Needleman and C.D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of molecular biology*, vol. 48, no. 3, pp. 443-453, 1970.

[11] S. Henikoff and J.G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, p. 10915, 1992.

[12] R. Durbin, "Biological sequence analysis: probabilistic models of proteins and nucleic acids," 1998.

[13] Wikipedia. Graphics processing unit. [Online]. http://en.wikipedia.org/wiki/Graphics_processing_unit

[14] Ali Akoglu Greg Streimer, "Sequence Alignment with GPU: Performance and Design Challenges," in *23rd IEEE International Parallel and Distributed Processing Symposium*, Rome, Italy, May 25-29, 2009.

[15] HT Kungt and C.E. Leisersont, "Systolic arrays (for VLSI)," in *Society for Industrial \& Applied*, 1979, p. 256.

[16] CW Yu, KH Kwong, KH Lee, and PHW Leong, "A Smith-Waterman systolic cell," *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pp. 291-300, 2005.

[17] J. Allred et al., "Smith-Waterman implementation on a FSB-FPGA module using the Intel Accelerator Abstraction Layer," in *IEEE*, 2009, pp. 1-4.

[18] P. Zhang, G. Tan, and G.R. Gao, "Implementation of the Smith-Waterman algorithm on a reconfigurable supercomputing platform," in *ACM*, 2007, pp. 39-48.

[19] L. Wienbrandt, S. Baumgart, J. Bissel, C.M. Yen Yeo, and M. Schimmler, "Using the reconfigurable massively parallel architecture COPACOBANA 5000 for applications in bioinformatics," *Procedia Computer Science*, vol. 1, no. 1, pp. 1021-1028, 2010.

[20] XtremeData. XtremeData XD2000i in-socket accelerator product flyer. [Online]. http://www.xtremedata.com/images/pdf/xd2000i_product_flyer.pdf

[21] High Performance Computing and Networking UAM. [Online]. http://www.hpcn.es/

[22] Impulse-C. [Online]. http://www.impulseaccelerated.com

[23] H. Abelsson, G. Sandberg, and S. M, "Accelerating NCBI BLAST".

[24] Smith-Waterman Wikipedia. [Online]. http://en.wikipedia.org/wiki/Smith_waterman

[25] E. Sotiriades and A. Dollas, "A general reconfigurable architecture for the BLAST algorithm," *The Journal of VLSI Signal Processing*, vol. 48, no. 3, pp. 189-208, 2007.

[26] S. Kasap, K. Benkrid, and Y. Liu, "Design and implementation of an FPGA-based core for gapped BLAST sequence alignment with the two-hit method," *Engineering Letters*, vol. 16, pp. 443-452, 2008.

[27] K. Muriki, K.D. Underwood, and R. Sass, "RC-BLAST: Towards a portable, cost-effective open source hardware implementation," in *IEEE*, 2005, p. 8.

[28] M.C. Herbordt, J. Model, B. Sukhwani, Y. Gu, and T. VanCourt, "Single pass streaming BLAST on FPGAs," *Parallel computing*, vol. 33, no. 10-11, pp. 741-756, 2007.

[29] D.T. Hoang, "Searching genetic databases on Splash 2," in *IEEE*, 2002, pp. 185-191.